Modélisation Géométrique par Contraintes : Solveurs basés sur l'arithmétique des intervalles

Abdou El Karim Tahari

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Supérieure d'Informatique

THESE

Présentée
En vue de l'obtention du titre de

Docteur en Sciences Informatique

Par

Abdou El Karim TAHARI

Intitulé :

Modélisation Géométrique par Contraintes : Solveurs basés sur l'arithmétique des intervalles

Soutenue publiquement le : 29/09/2011

Devant le jury composé de:

M. Amar Balla	Professeur à L'ESI	Président
M. Mohamed Ahmed Nacer	Professeur à L'USTHB	Examinateur
M. Hamid Hadadou	M.C.A à L'ESI	Examinateur
M. Abdel wahab Moussaoui	M.C.A à L'U.F.A.Sétif	Examinateur
M. Mohamed Bachir Yagoubi	M.C.A à L'U.A.T.Laghouat	Examinateur
M. Samy Ait Aoudia	Professeur à L'E.S.I	Directeur de thèse

Remerciements

Mes premiers remerciements vont à Monsieur Samy. Ait Aoudia, mon Encadreur, parce qu'il m'a fait découvrir le thème de synthèse d'images, et je lui en suis très reconnaissant. Il a su me motiver et m'accompagner dans ce projet passionnant où il y a beaucoup à faire.

Je tiens à remercier vivement Monsieur Sebti Foufou, Professeur à l'Université de Bourgogne (Dijon), pour m'avoir encouragé et m'avoir aidé dans mes débuts difficiles. Ses qualités, tant humaines que scientifiques furent pour moi d'un apport inestimable. Je lui en suis très reconnaissant.

Je tiens à exprimer ma profonde gratitude et mes remerciements les plus sincères à Monsieur Dominique Michelucci, Professeur à l'Université Bourgogne, Dijon, pour la confiance qu'il m'a toujours témoignée. J'ai tout particulièrement apprécié ses encouragements et ses conseils, surtout en fin de thèse. Notre collaboration m'a permis de progresser et de me constituer de solides bases pour le futur.

Je voudrais aussi exprimer mon profond respect et mes remerciements à Monsieur Adel Moussaoui qui à donné une autre dimension à mon travail et avec qui j'ai travaillé en etroite collaboration sous la direction de Monsieur Samy Ait Aoudia.

Je remercie les responsables du laboratoire LE2I de Dijon de m'avoir fait confiance en m'accueillant au sein de leur laboratoire et où ils donnent à mon travail un autre aspect.

Je voudrais aussi exprimer ma gratitude au président de jury et à tous les membres du jury pour l'honneur qu'ils m'ont fait en acceptant de juger ce travail.

Je n'oublie pas d'adresser mes vifs remerciements à tous mes collègues, du département de l'informatique et de l'Université Amar Telidji qui m'ont aidé de près ou de loin dans l'accomplissement de ce travail. Je n'ose citer des noms de peur d'oublier quelqu'un mais je garderai en mémoire leur aide et leur soutien.

Table des matières

Table des figures

Introduction

Plusieurs applications de l'informatique graphique ou géométrique nécessitent la résolution de systèmes non linéaires. Le calcul des images de synthèse par lancer de rayons nécessite le calcul des points d'intersection entre des rayons (des demi-droites) et des surfaces définies par une ou plusieurs équations algébriques. La modélisation géométrique, et la Conception et Fabrication Assistées par Ordinateur (CFAO) ont besoin de calculer les points d'intersection entre de telles surfaces. Il en résulte des systèmes d'équations algébriques de petites tailles. Enfin, tous les modeleurs géométriques utilisés en CFAO fournissent aujourd'hui la possibilité de modéliser des objets géométriques, ou de dimensionner des pièces, par un ensemble de contraintes géométriques, telles que des relations d'incidences, de tangences, des spécifications d'angles ou de distances entre des éléments géométriques : points, droites, plans, cercles, sphères, etc [2]. La résolution de ces contraintes géométriques nécessite la résolution de systèmes d'équations algébriques non linéaires. Les sous systèmes irréductibles peuvent être de grande taille (plus d'une dizaine d'inconnues et d'équations) et sont résolus par des méthodes numériques : citons l'itération de Newton-Raphson, l'homotopie (ou continuation), les méthodes de Newton par intervalles [3], [4], [5], et les solveurs utilisant les bases tensorielles de Bernstein ou d'autres bases géométriques [6] [7], [8], [9], [10], [11].

Les bases tensorielles de Bernstein sont utilisées en informatique graphique pour calculer de bons encadrements des valeurs d'un polynôme multivarié $f(x)$, $x = (x_1, \ldots x_n)$, sur l'hypercube $x \in [0,1]^n$ ou sur un pavé $u_i \leq x \leq v_i$ de R^n : le polynôme $f(x)$ est exprimé dans la base tensorielle de Bernstein, et $f([0,1]^n)$ est contenu dans l'intervalle défini par le plus petit et le plus grand de ces coefficients. Contrairement aux coefficients dans la base habituelle : $(1, x_1, x_1^2, \ldots) \times (1, x_2, x_2^2, \ldots) \times (1, x_3, x_3^2, \ldots) \times \ldots$, dite base canonique, les coefficients dans la base tensorielle de Bernstein dépendent du pavé contenant l'argument x. Un pavé est décrit par un couple $[u, v]$ où $u \in R^n$ et $v \in R^n$, et est l'ensemble des points $x = (x_i)$ de R^n tels que $u_i \leq x_i \leq v_i$. La méthode de Paul de Casteljau permet de calculer ces coefficients pour un sous pavé, sans repasser par la base canonique. Tout cela est aujourd'hui classique en informatique graphique ; par exemple les bases de Bernstein sont utilisées de façon routinière pour calculer des couvertures de courbes ou de surfaces algébriques implicites [12]. Récemment, les propriétés des bases tensorielles de Bernstein, ou d'autres bases géométriques (splines rationnelles) ont été utilisées, au delà du 2D et du 3D, pour des solveurs de systèmes d'équations algébriques non linéaires [7], [8], [9], [10], [11].

Toutefois, les polynômes qui sont habituellement creux dans la base canonique deviennent denses dans la base tensorielle de Bernstein. Par exemple, le monôme 1

s'écrit ($B_0^{(d_1)}(x_1) + \dots\ B_{d_1}^{(d_1)}(x_1)) \times \dots \times (\ B_0^{(d_n)}(x_n) + \dots\ B_{d_n}^{(d_n)}(x_n))$) dans la base tensorielle de Bernstein, où $B_i^{(d_k)}(x_k) = \binom{d_k}{i} x_k^i (1-x_k)^{d_k - i}$ est le $i$ème polynôme dans la base des polynômes de Bernstein de degré d_k. De même, un polynôme linéaire $p(x_1, \dots x_n)$ nécessite un nombre exponentiel 2^n de coefficients dans la base tensorielle de Bernstein, alors qu'il suffit de $n+1$ coefficients dans la base canonique. Un polynôme quadratique nécessite 3^n coefficients dans la base tensorielle de Bernstein, et $O(n^2)$ coefficients dans la base canonique, pour les monômes : x_i^2, $x_i x_j$, x_i et 1.

Ceci rend les solveurs classiques de Bernstein inutilisables pour les systèmes de plus de 6 ou 7 équations. Or les contraintes géométriques, surtout en 3D, fournissent des systèmes irréductibles bien plus grands. Par exemple un icosaèdre régulier, ou non régulier (20 triangles, 12 sommets, 30 arêtes), peut être spécifié par la longueur de ses arêtes, ce qui donne un système non réductible de 30 équations ; de même leur dual, le dodécaèdre régulier ou non (12 faces pentagonales, 20 sommets, 30 arêtes) peut être spécifié par les longueurs de ses 30 arêtes et les coplanarités de ses 12 faces pentagonales. Ces systèmes sont quadratiques. En utilisant les notations habituelles, les équations sont : $a_k^2 + b_k^2 + c_k^2 = 1$, $a_k x_i + b_k y_i + c_k z_i + d_k = 0$, $(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 = D_{ij}^2$, où (x_i, y_i, z_i) sont les coordonnées du sommet i, où $a_k x + b_k y + c_k z + d_k = 0$ est l'équation du plan de la face numéro k, et D_{ij} est la longueur de l'arête ij. Pour obtenir un système bien contraint, 3 sommets peuvent être fixés, par exemple un sommet est fixé à l'origine, un premier voisin est fixé sur l'axe des x, et un second est fixé dans le plan xy. Ces systèmes sont gros, et quasiment irréductibles. Ils ne peuvent être résolus par les solveurs de Bernstein classiques, à cause de la taille exponentielle de la représentation des polynômes dans la base tensorielle de Bernstein.
Afin de remplir ces objedtifs, ce document se divise en plusieurs chapitres.

Le chapitre 1 constitue d'une certaine manière un état de l'art dans le domaine de la modélisation géométrique par conraintes ; il définit les notions importantes utilisées tout au long de cette étude. Ce chapitre présente les différentes familles des méthodes de la modélisation, ainsi que les différentes méthodes de résolution.

Le chapitre 2 présente l'arithmétique par intervalles et quelques variantes limitant la surestimation.

Le chapitre 3 méthode utilisant la base de Bernstein couplée à l'algorithme de De Casteljau pour encadrer des polynômes d'une manière trés fine.

Le chapitre 4 propose un nouvel algorithme polynomial pour résoudre la difficulté relative à la taille de la représentation des polynômes dans la base tensorielle de Bernstein. Jusqu'à maintenant, la seule solution consistait à ne pas utiliser la base tensorielle de Bernstein, mais la base simplicielle (dite aussi homogène) de Bernstein [13], [10]. Dans ce chapitre on décrit les solveurs classiques fondés sur les bases de Bernstein ; ces solveurs ne peuvent traiter que des petits systèmes, avec 6 à 7 inconnues au plus, comme il en apparaît en synthèse d'images, par exemple pour le

lancer de rayons sur des surfaces paramétriques.

Le chapitre 5 on présente un nouveau type de solveur : il évite cette limitation en définissant le polytope de Bernstein et en recourant à la programmation linéaire [14]. Ce deuxième type de solveurs est utilisable sur des systèmes de taille arbitraire : la réduction d'un pavé, en préservant la ou les racines contenues, est effectuée en temps polynomial, si bien que chaque racine réelle simple est isolée en temps polynomial. Bien sûr, ce solveur nécessite toujours un temps exponentiel quand le pavé étudié contient une quantité exponentielle de racines réelles.

Chapitre 1

La modélisation géométrique par contraintes

1.1 Introduction

La modélisation géométrique [15] [16] joue un rôle important dans le processus de conception des produits industriels. Dans les applications CAO et DAO, l'utilisateur fourni explicitement l'emplacement des objets géométriques et spécifie les différentes relations que doivent vérifier les différentes parties de l'objet, on les appelle contraintes géométriques ; on peut citer comme types de contraintes géométriques classiques en 2D pour spécifier les éléments tels que les points, les droites et les cercles, les contraintes de distance entre points, distance entre points et droites, angles entre droites, parallélisme entre droites, incidence entre points et droites, entre points et cercles, tangence entre cercles, tangence entre droites et cercles, etc. L'utilisateur peut spécifier des contraintes redondantes ou contradictoires, on dira que l'objet est sur-contraint ou sur-déterminé. Si par contre les contraintes sont insuffisantes, on dira que l'objet est sous-contraint ou sous-déterminé.
Ce type de modélisation conduit à des systèmes d'équations vite compliquées. En particulier, il n'est pas évident pour l'utilisateur de savoir si tel système est bien contraint, sur contraint ou sous-contraint, car la spécification des contraintes est une tâche fastidieuse et largement exposée aux erreurs. Il est crucial d'aider les utilisateurs à déboguer leurs systèmes de contraintes ; un système de diagnostic ou d'analyse qualitative des contraintes est indispensable pour atteindre cet objectif. Dans un système de CAO piloté par les contraintes, l'utilisateur trace une esquisse composée d'un ensemble d'éléments géométriques et de contraintes. Le système s'occupe de la résolution et de la vérification de l'esquisse proposée. Il avertit l'utilisateur en cas de manque, redondance ou contradiction. Il n'est pas suffisant que l'étape de résolution se limite à avertir l'utilisateur s'il y a une erreur car il veut un diagnostic plus précis en indiquant les parties sur ou sous-contraints. Le processus de dessin est intuitif pour l'utilisateur même s'il n'est pas conscient des mécanismes internes et aspects technique du système. L'objet conçu peut être utilisé comme une partie utile d'un objet plus grand, il peut être utilisé dans des phases subséquentes d'une application(par exemple mouvement d'un robot), ce qui implique que les informations reliées aux intentions de l'utilisateur doivent être préservées.

Qu'est ce qu'une contrainte ? [15] [16]
Une contrainte est généralement une relation désirée entre plusieurs inconnues (ou variables), chacune prend une valeur dans un domaine donné. Une contrainte restreint donc les valeurs que les variables peuvent prendre, elle représente de l'information partielle.
Contrainte géométrique [15] [16]
Une contrainte géométrique est une relation entre deux objets ou entités géométriques. On peut citer par exemple : distance entre deux points, angle entre deux droites, incidence entre un point et une droite, tangence entre cercles.
Objet sur-contraint [15] [16]
Si les contraintes données par l'utilisateur pour définir l'objet sont redondantes ou contradictoires, on dira que l'objet est sur-contraint ou surdéterminé.
Objet sous-contraint [15] [16]
Si les contraintes sont insuffisantes (une infinité non dénombrable de solutions), on dira que l'objet est sous-contraint ou sous-déterminé.
Objet bien-contraint : [15] [16]
S'il a un nombre fini de solutions.

FIGURE 1.1 – Exemple d'un carénage de moto

La figure 1.1 propose l'exemple réel d'un carénage de moto et son modèle défini par une surface B-spline percée par deux trous. Le maillage des points définissant cette surface B-spline est également visible, et souligne la difficulté pour définir les lieux des points de contrôle. Ce chapitre a pour but de montrer certaines de ces bases et est organisé comme suit.
La partie 1.1 est consacrée à une présentation des différentes méthodes de modélisation par contraintes existantes.
La partie 1.2 présente les méthodes de résolution et de décomposition de contraintes en CAO.

1.2 Différentes méthodes de modélisation

Plusieurs méthodes ont été étudiées et développées dans ce cadre ; dans notre cadre d'étude on peut décrire deux grands axes. [15] [16]

– les méthodes paramétriques et variationnelles.

– les méthodes par attributs (*features*)

.

R.Maculet et M.Daniel dans [17] suggérent que la première distinction qui peut être faite entre ces différentes approches est le niveau d'abstraction employé pour manipuler un modèle.

– **Le niveau 0**
Des paramètres, ou des variables seront manipulées. Par exemple, un point est défini par deux paramètres (x, y) en dimension deux, et par trois paramètres (x, y, z) en dimension trois.
– **Le niveau 1**
Des objets géométriques élémentaires seront manipulés (points, droites, courbes, surfaces) ; il correspond aux modeleurs paramétriques et variationnels capables de résoudre des contraintes géométriques élémentaires (distance entre deux points, angle entre deux droites, etc.)
– **Le niveau 2**
Des objets géométriques composés d'éléments plus simples du niveau 1 combinés entre eux (par exemple, une rainure sur un objet) ; ce niveau correspond aux méthodes par attributs, pour résoudre des contraintes complexes (comme la longueur de la rainure) qui sont généralement associées à une signification sémantique ou des propriétés géométriques.

Le deuxième point permettant de différencier les modeleurs est le concept de contrainte orientée et de contrainte non orientée. Par exemple, soit un rectangle de longueur l et de largeur L. Une contrainte orientée pourrait être du type : $l := 2 \times L$. La largeur L est initialement connue, la longueur l peut être ensuite évaluée grâce à cette contrainte orientée. Par ailleurs, une contrainte non orientée est une equation telle que : $l - (2 \times L) = 0$. Dans ce cas, les grandeurs l ou L peuvent être modifiées indifféremment.

La troisième différence est d'exprimer soit des contraintes géométriques, soit des contraintes incluant des paramètres non géométriques. Les contraintes d'ingénierie les plus utilisées désignent des équations contenant des paramètres géométriques de l'objet aussi bien que des paramètres technologiques et mécaniques (forces, puissance, efforts, etc.).

Plusieurs méthodes existent actuellement pour satisfaire les besoins du concepteur, selon le niveau d'abstraction désiré et considéré pour manipuler un objet. Dans ce qui suit nous allons aborder : les méthodes paramétrique et variationnelle d'une

part et les méthodes par attributs et déclaratives d'autre part.

1.2.1 Les méthodes paramétriques

A ce niveau, uniquement les éléments des deux premiers niveaux seront pris en compte, Un objet donné par des caractéristiques géométriques (longueur, hauteur, angles, etc.) peut être défini et obtenu par l'indication de paramètres ou de variables [15] [16].
Ce sont des méthodes séquentielles car elles permettent la modification d'un modèle par changement des instanciations de ses parties géométriques constitutives, données dans un ordre précis. Par exemple, un modeleur paramétrique peut construire un triangle défini par deux côtés et un angle (cf. Figure 1.2, gauche) ; le processus de génération est effectué séquentiellement : le point A est placé, puis la droite horizontale est tracée avec le point B placé à la distance dAB ; puis la droite qui forme

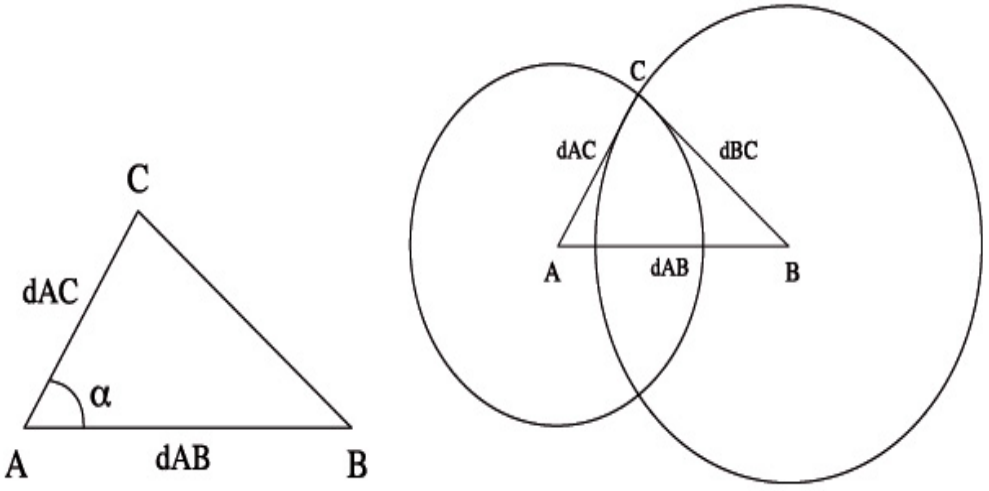

FIGURE 1.2 – A gauche :construction d'un triangle défini par deux distances dAB et dAC, et un angle α ; à droite : le problème est cyclique en considérant simultanément les distances dAB etdBC

un angle α avec dAB est tracée et finalement le point C est placé à la distance dAC. D'autre part, l'exemple suivant d'un triangle donné par les longueurs de ses trois côtés seulement ne peut être résolu séquentiellement. Dans ce cas, les distances dAB et dAC doivent être considérées simultanément pour résoudre le problème (cf. Figure 1.2, droite). Ce sont deux contraintes couplées et le problème est dit cyclique. Une approche variationnelle est ainsi nécessaire (cf. partie 1.2.2) pour obtenir une solution.
Le problème à résoudre est dit paramétrique s'il peut être divisé en un ensemble de sous-problèmes pouvant être résolus séquentiellement, l'un après l'autre [18]. Il y a ainsi un historique de l'ordre de construction qui doit pouvoir être étendu à chaque modification du modèle. Puisqu'il n'est pas possible de résoudre un problème général, ni de placer des objets simultanément, cette approche présente des limitations

sur les contraintes à considérer. Ces contraintes, à partir desquelles tous les paramètres peuvent être extraits, sont utilisées pour générer une solution.
Les limites de la modélisation par ces méthodes sont dues à la faiblesse de la résolution des contraintes ; il n'y a aucune association possible entre les contraintes géométriques et les contraintes d'ingénieries [19]. Chaque étape du processus de résolution ne tient compte que d'une contrainte impliquant des objets géométriques à la fois : cette contrainte est au «niveau 1 » et correspond à un sous-système d'équations au niveau des variables, de «niveau 0». Les contraintes ne doivent pas :

- Etre cycliques comme dans l'exemple précédent de construction d'un triangle pour lequel seules les longueurs des trois côtés sont connues ;
- Correspondre à des constructions du type « à la règle et au compas» [20].

Ce sont des systèmes irréductibles.
Les méthodes paramétriques donnent une solution unique si elle existe et sont incapables de fournir des solutions multiples. L'utilisateur n'a accès qu'à un ensemble prédéterminé de paramètres qu'il doit implémenter dans un ordre prédéfini, réduisant ainsi la flexibilité du processus de conception. D'un autre point de vue, si un paramètre n'est pas adapté ou hors de son domaine de validité, la séquence des opérations peut donner une indication sur l'influence de chaque paramètre, chose qui pourrait être plus compliquée à obtenir avec la prochaine catégorie de méthodes.
Ajoutons à cela que l'ensemble des contraintes exprimées doit être impérativement bien contraint. Cependant, pendant le processus de conception, cet ensemble est généralement sous-contraint et parfois sur-contraint, car il n'est pas toujours évident pour le concepteur de s'assurer que toutes les contraintes restent cohérentes et que le nombre de variables introduites puisse décrire correctement les variations de l'objet. L'ensemble des contraintes devient bien contraint après que le concepteur ait soigneusement analysé les variations qu'il désire apporter à l'objet. C'est dans ce but que les méthodes variationnelles ont été développées, afin de donner une meilleure réponse aux besoins du concepteur.

1.2.2 Les méthodes variationnelles

La dénomination de la géométrie variationnelle est apparue la première fois en CAO avec Lin, Gossard et Light [21]. Cette approche est complètement déclarative si on considère uniquement des entités des deux premiers niveaux.
Hoffmann et Joan-Arinyo ont établi dans [18] qu'un problème est variationnel s'il est divisible en plusieurs sous-problèmes qui peuvent être résolus simultanément comme des systèmes de contraintes. Les contraintes peuvent être géométriques aussi bien que d'ingénierie. Cependant, ces dernières sont souvent des équations contenant des paramètres géométriques et non géométriques de l'objet, ce qui induit généralement une relation forte entre ces deux types de contraintes.
Suivant la configuration du problème (sous, bien, ou sur-contraint), le solveur est capable de trouver un ensemble correct de solutions. Pour les configurations dans lesquelles aucune solution n'est trouvée, deux cas se présentent : soit le problème est sur-contraint et n'a aucune solution quelles que soient les valeurs des parametres, soit il est bien contraint mais n'a pas de solution pour ces valeurs, par exemple une inégalité triangulaire est isolée. Plusieurs méthodes de décomposition ont été

développées pour suppléer les méthodes de résolution et pour aider le concepteur rencontrant des configurations sous ou sur-contraintes.
Cette résolution globale d'un ensemble de contraintes est plutôt complexe ; la complexité de la résolution et le temps de calcul augmentent à mesure que le nombre d'objets géométriques devient plus important. La plupart des modeleurs variationnels travaillent en dimension deux par opposition aux modeleurs paramétriques qui permettent de manipuler des configurations en trois dimensions. Ces modeleurs variationnels s'appellent des «sketchers» : le concepteur travaille interactivement avec les systèmes de C.A.O, et produit une esquisse préliminaire de l'objet.

1.2.3 Les méthodes par attribut de forme

Ces méthodes considèrent des entités du deuxième niveau, appelées attributs («feature», ou caractéristiques) qui peuvent être de plusieurs types, et en particulier du type attributs de forme afin de caractériser la forme composant un objet. En outre, des entités complémentaires ont été intégrées dans le modèle précédent, permettant la caractérisation de quelques opérations du processus de fabrication ou de toute autre connaissance du concepteur à d'autres étapes du processus de conception. Le but de ces méthodes est de pouvoir manipuler directement la forme d'un objet et la sémantique associée à la place de la définition purement géométrique de l'objet. Ces entités sont des ensembles complexes d'éléments simples : par exemple, la caractéristique «trou» est composée d'un ensemble de cylindres et de plans rattachés à un plan initial. L'extension des attributs au domaine de formes libres est restée un problème bien connu pendant une quinzaine d'années. A titre illustratif, la forme du carénage de moto (cf. Figure 1.1) peut être conçue avec une méthode basée sur les attributs de forme, avec par exemple une caractéristique «trou» pour l'ouverture du système de refroidissement et une caractéristique «bosse» pour l'aileron (cf. Figure 1.3).
Une classification a été proposée par Fontana [22] et prolongée par Perrot [23].

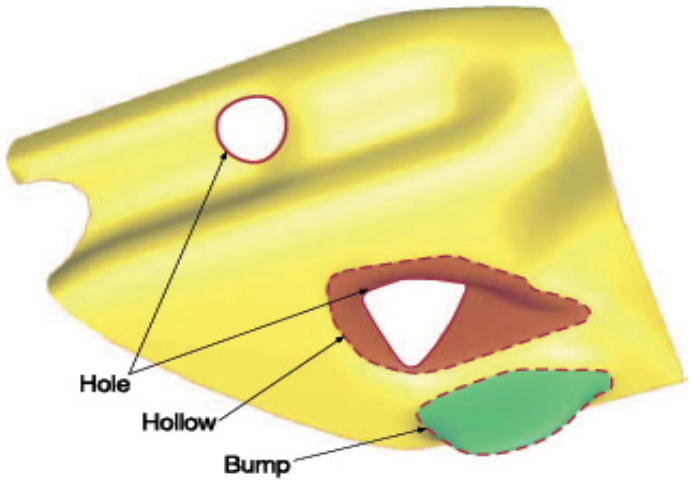

FIGURE 1.3 – Le carénage de moto conçu par attributs

Quatre niveaux ont été proposés, allant d'un bas niveau de contrôle de forme à des niveaux plus élévés (cf. Figure 1.4)) :

- Les **attributs de formes** sont composés simplement de surfaces primitives, telles que des plans ou des cylindres. Quelques attributs de formes sont disponibles dans les modeleurs de CAO actuels, comme la caractéristique «trou» vue précédemment.
- Les **attributs de formes semi-libres** sont définis par des surfaces de formes libres, et obtenus par des règles de génération de surfaces classiques telles que les règles d'interpolation ou les relations spécifiques exprimées directement entre les points de contrôle. De telles surfaces sont ainsi totalement définies par un ensemble restreint de paramètres, en argumant des opérations de génération.

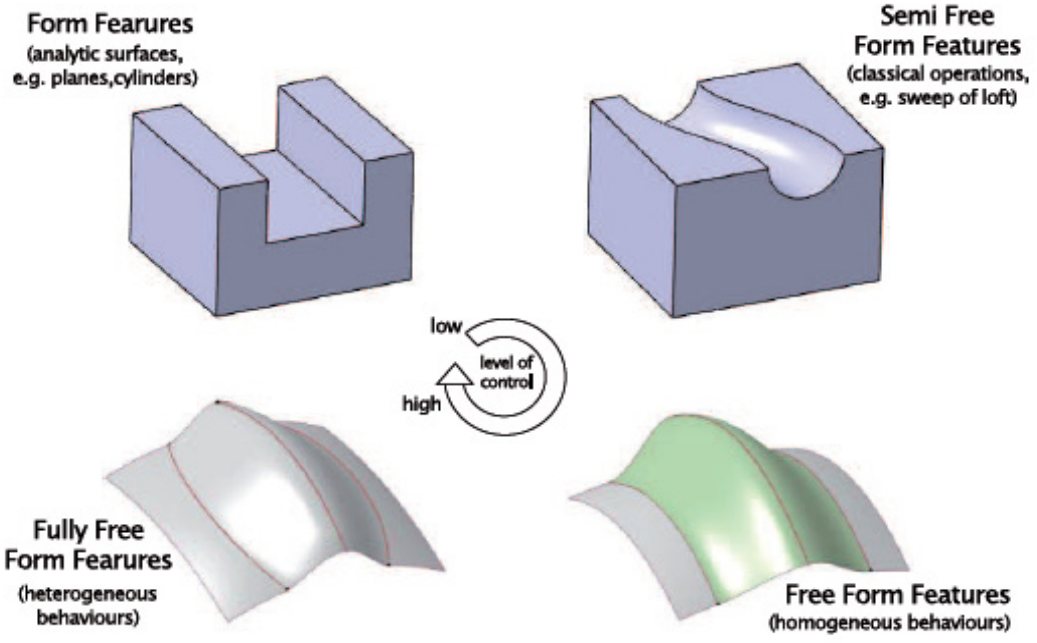

FIGURE 1.4 – Classification des attributs basée sur le niveau de contrôle de la forme

- Les **attributs de forme libres** sont obtenus par l'utilisation des techniques de changement de formes, souvent basées elles mêmes sur des techniques de déformation imposant un comportement homogène à toute la surface. Cela permet d'obtenir plus de liberté dans la définition d'une forme complexe, mais le suivi de la zone modifiée n'est pas toujours évident.
- Les **attributs de formes totalement libres** sont caractérisés par un niveau de liberté supérieur dans la définition de la forme et obtenus par l'utilisation de techniques autorisant des comportements hétérogènes sur différentes zones de la surface de forme libre.

.
Les méthodes par attributs peuvent dériver des méthodes paramétriques (un ensemble de paramètres prédéterminés définit complètement la dimension intrinsèque d'un attribut, et sa position sur l'objet) ou des méthodes variationnelles (un ensemble d'équations ou de contraintes peut le définir).
Les premiers travaux sur ce thème sont de Shah [24] et Marks [25] ; il y a deux types d'approches pour les méthodes par attributs [26] : l'approche procédurale qui est semblable aux méthodes paramétriques en ce qui concerne le processus de résolution, et l'approche déclarative semblable aux méthodes variationnelles. Mais la majorité des systèmes de CAO industriels basés sur des méthodes de modélisation par attributs sont paramétriques. Ils sont par conséquent de types procéduraux et dépendent de la séquence d'instruction pour générer ces attributs.
Dans le domaine des surfaces de forme libre, les méthodes de modélisation variationnelle et par attributs ne sont pas encore bien établies. En raison de leur complexité, il est dans un premier temps difficile de trouver un ensemble de paramètres correct pour décrire une forme libre, suivant la sémantique qui y est rattachée et le contexte de l'utilisateur. Deuxièmement, la traduction de ces paramètres en contraintes géométriques sur l'objet est une étape difficile, en particulier parce que le nombre et le type de paramètres ne sont pas adaptés aux degrés de liberté du modèle géométrique.
Une manière de créer et de manipuler facilement des attributs de forme libre est de soumettre un processus de déformation de forme libre à des contraintes [23] : un ensemble de contraintes est appliqué à la surface et le processus de déformation est combiné à une fonction à optimiser afin de choisir une solution. Dans ce cas, les paramètres d'un attribut de forme libre sont directement liés aux paramètres du processus de déformation.

1.2.4 Les méthodes déclaratives

L'approche déclarative peut être vue comme une extension des méthodes par attributs, bien qu'il n'y ait aucun lien historique entre elles. Le niveau d'abstraction est ici plus élevé, puisque les méthodes déclaratives sont placées au niveau de la modélisation conceptuelle ou sémantique, équivalent à un «niveau 3» [27].
Le but de la modélisation déclarative est de permettre à l'utilisateur de créer le modèle numérique d'un objet physique, en ne donnant qu'un ensemble de propriétés. L'ordinateur établit une, quelques ou toutes les solutions correspondant à la description de l'utilisateur ; idéalement, le concepteur est ainsi libéré de tout calcul et peut se concentrer sur la phase de création. Ces méthodes sont destinées à esquisser rapidement un croquis ou les contours de la forme de l'objet. La conception détaillée est a priori plus aisée avec un modeleur classique ; dés lors que toutes les propriétés de l'objet ont été données, il est possible d'avoir à la fois une description géométrique et sémantique. Néanmoins, aussi détaillé que puisse être le modèle de l'objet, le fait de maintenir la cohérence entre le modèle sémantique et le modèle géométrique en tenant compte d'éventuelles modifications géométriques de la part du concepteur reste un défi.
Différentes applications ont été proposées avec cette approche : le polyèdre [28], le

contrôle spatial [29], ou encore la modélisation de scènes de synthèse d'images [30], [31]. Dans notre contexte, des recherches ont été réalisées sur les courbes [32], [33] et les surfaces , [34]. L'exemple proposé dans la figure 1.5 montre les frontières entre les zones où les différentes contraintes doivent être appliquées (à gauche, dans le domaine paramétrique ; à droite, sur les surfaces résultantes)

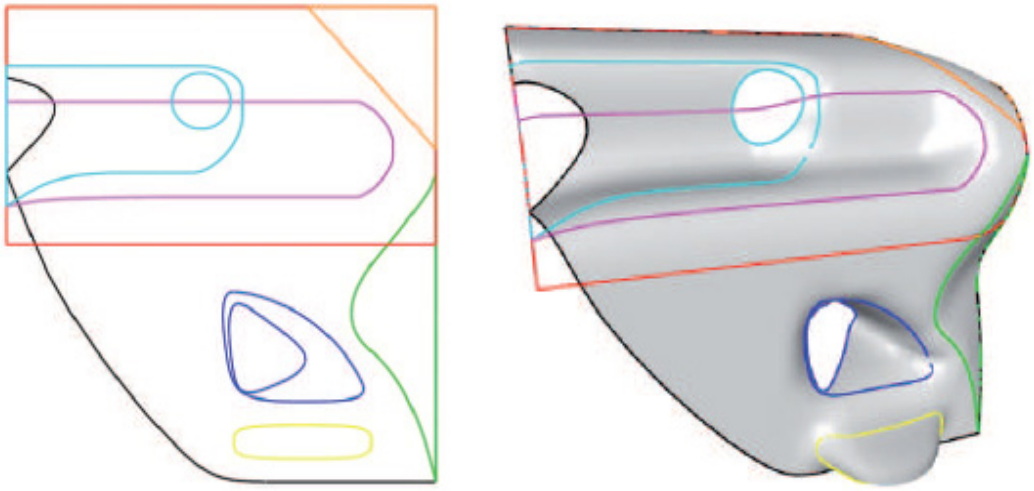

FIGURE 1.5 – Limites des zones pour la modélisation du carénage de moto : à gauche, dans le domaine paramétrique ; à droite, sur la surface

Un système global de modélisation déclarative doit inclure plusieurs outils et fonctions [35] :

- Pour l'étape de la description, durant laquelle le modèle conceptuel est explicité de manière déclarative, et affiné par une boucle de conception/reconception. Plusieurs méthodes de description existent déjà : introduction de la logique floue [36], utilisation du langage naturel ou pseudo-naturel [33], schémas et diagrammes en 2D pour la conception préliminaire , etc.
- Pour l'étape de génération, au cours de laquelle la résolution d'un système de contraintes résulte des propriétés établies durant la description.
- Un outil permettant à l'utilisateur de parcourir l'ensemble des solutions et lui laissant la possibilité de choisir une ou plusieurs solutions, puis de reprendre le processus dans la boucle de conception/reconception.

.

1.3 Méthodes de résolution et de décomposition de contraintes en CAO

A l'heure actuelle, [37], [38] la modélisation par contraintes, en particulier avec le variationnel, aboutit à de grands systèmes d'équations. Pour aider le concepteur qui travaille en interactif, il faut que la résolution soit efficace.

Les systèmes d'équations utilisés en CAO sont quelconques : linéaires, non linéaires polynomiales, algébriques ou transcendantes, ce qui accroit la difficulté de la résolution.
Pour résoudre simultanément un grand système d'équations représentant des contraintes très interdépendantes, il est nécessaire de faire une décomposition en des sous-systèmes les plus indépendants possible, d'où le concept de PPSI (Plus Petit Système Indépendant) [20]. Dans un problème industriel, le nombre d'équations et de variables se compte en centaines voire en milliers. Nous allons présenter différentes méthodes de résolution puis de décomposition utilisées ou essayées en CAO.

1.3.1 Les méthodes de résolution

D'aprés [37], il existe plusieurs méthodes de résolution

1.3.1.1 Les méthodes formelles

Les méthodes formelles [37] aboutissent à des résultats exacts mais sont trop lentes pour être utilisables en C.A.O (temps exponentiel). Elles sont nombreuses : la méthode des bases de Grbner [39], la méthode de Ritt-Wu [40], [41], la méthode des résultants [42] ; on a aussi des méthodes à base de règles géométriques [43] comme, par exemple les méthodes de construction formelle à la règle et au compas. Le logiciel de calcul formel Mapple a été utilisé [44]. On peut classer parmi ces méthodes l'approche de Martin [31], [45].

1.3.1.2 Les méthodes numériques itératives

Les méthodes numériques [37], sont les plus employées car elles sont rapides (temps polynomial). Par contre elles ne sont pas exactes. Elles sont adaptées aux systèmes de grande taille de la CAO. Pour ces raisons, certaines de ces méthodes sont aujourd'hui utilisées dans les logiciels de CAO industriels.
Les méthodes classiques pour résoudre ces systèmes d'équations non linéaires sont itératives. Il est absolument nécessaire avant d'appliquer une des méthodes d'avoir une solution approchée. Ce n'est d'ailleurs pas un inconvénient en CAO puisque le concepteur part toujours d'une solution. A cette étape de la résolution, cette solution n'est pas encore assez contrainte ; le système est sous-contraint. Le concepteur ajoute ensuite des contraintes.
La méthode la plus employée est une méthode bien connue en analyse numérique : la méthode de Newton-Raphson ou une variante [21], [46], [47]. Cette méthode a un inconvénient : elle converge parfois mal. La méthode par homotopie [48], [49] converge d'une manière plus continue, plus prévisible, moins chaotique que la méthode Newton-Raphson.La méthode de résolution par intervalles est issue du mariage de l'analyse par intervalles et des techniques de programmation par contraintes [32], [33]. Cette approche basée sur le calcul d'intervalles est intéressante car elle permet d'effectuer du calcul numérique exact : l'arithmétique d'intervalles définit tous les opérateurs calculatoires sur les intervalles de façon à ce qu'aucune solution ne soit

perdue. La résolution par intervalles combine donc les avantages des méthodes numériques et des méthodes formelles. Ces méthodes sont plus ou moins compatibles avec les méthodes de décomposition.
Bien sûr ces méthodes ne sont pas complètes. Ainsi la méthode de Newton-Raphson n'obtient pas toutes les solutions. Remarquons enfin que la résolution d'un système d'équations non linéaire peut être ramenée au calcul d'un minimum d'une fonction. Nous donnons dans les chapitres suivants les fondements techniques de la méthode : arithmétique d'intervalles, découpage et résolution par intervalles.

1.3.2 Les méthodes de décomposition

Ces méthodes [37] sont classées selon deux grandes catégories : les méthodes de décomposition équationnelles et les méthodes de décomposition géométriques [21], les méthodes équationnelles se situent au niveau variables-équations, c'est à dire au «niveau 0» et donc décomposent des CSP (Constraint Satisfaction Problems) quelconques. Les méthodes géométriques sont en niveau des objets géométriques et des contraintes, c'est à dire au «niveau 1» et s'appliquent donc aux GCSP (Geometric CSP)

1.3.2.1 Les méthodes de décomposition équationnelles

Ces méthodes [37] s'appliquent au graphe variables-équations, au «niveau 0». Le système d'équations est décomposé en sous-systèmes sous, bien ou sur-contraints. Il existe plusieurs méthodes : de Lathan et Middledith [43], de Hsu et Brudelin [50], S.Ait-oudia, R.Jegou, D.Michelucci [51].

1.3.2.2 Les méthodes de décomposition géométriques

Ces méthodes [37] travaillent au niveau des objets géométriques et des contraintes, au «niveau 1». L'idée est d'identifier tous les systèmes bien-contraint ; plusieurs méthodes existent : l'approche descendante de Owen [21] D-Cubed Ltd avec le solveur DCM 2D, celle de Hoffmann, Lomonosov et Sitharam (HLS) .
Il est souhaitable de choisir une méthode de décomposition permettant une collaboration entre différentes méthodes de résolutions comme la méthode HLS. Ainsi chaque sous système décomposé est résolu par le solveur le plus approprié. Ainsi, on peut obtenir un modeleur CAO efficace car le temps de décomposition est polynomial et est donc négligeable par rapport au temps de résolution.

1.4 Conclusion

Nous avons fait un survol sur les méthodes les plus employées pour la résolution d'un problème de contraintes géométriques. Il n'existe pas de méthode universelle de résolution. La tentation initiale pour résoudre un système est de ramener le problème à la résolution d'un système numérique. L'approche purement algébrique

révèle trois inconvénients majeurs. Le premier est d'ordre numérique. En effet, le système d'équations n'est pas toujours aisé à résoudre, en particulier lorsque les équations non linéaires entrent en jeu. La résolution comporte certains inconvénients tels l'approximation de la solution, la dépendance vis à vis d'une valeur initiale ou encore le caractère discret des solutions. En confiant la résolution à un système de calcul formel, la qualité de la solution sera améliorée mais l'algorithme de Buchberger [21] est notoirement coûteux en temps. Les outils provenant de la recherche opérationnelle ou du calcul formel ne permettent donc pas de résolution précise, rapide et complète d'un système d'équations découlant d'un problème géométrique.
La seconde faiblesse apparait lors du traitement des problèmes sous-contraints. En effet, le concepteur ne connaît pas toujours entièrement les spécifications de son produit et doit constamment modifier son ensemble de contraintes. Se pose alors la question pour gérer l'ajout et le retrait de contraintes. Les approches algébriques ne présentent pas toutes de réponses adéquates.
Enfin, la dernière faiblesse est le fait que l'absence de raisonnement géométrique sur les contraintes exclut toute assistance au concepteur lors de la construction de la figure (déterminer les contraintes à relâcher lorsque le problème est sur-contraint, désigner les contraintes trop fortes rendant le problème incohérent, etc...). Nous voyons qu'avec les approches purement numériques, l'aspect géométrique est complètement ignoré, or le besoin d'un raisonnement géométrique se fait pourtant sentir.
Un avantage des méthodes numériques est qu'elles permettent de traiter un grand nombre de types de contraintes ; on peut en outre facilement ajouter de nouveaux types de contraintes.

Chapitre 2

Arithmétiques

2.1 Introduction

Avec l'accroissement de la vitesse des ordinateurs [52], [1], le calcul numérique a vu naître de nouvelles applications comme la modélisation de molécules ou encore la conception de modèles climatiques. Ces applications demandent de plus en plus une qualité et une précision de calcul accrues qui ne peuvent pas toujours être atteintes. La faute en incombe aux circuits et aux programmes arithmétiques qui peuvent comporter des erreurs (par exemple l'explosion d'Ariane 5 lors de son premier vol [ESA Report, 1996]) mais aussi aux nombres qui ne sont pas forcément représentables en machine et doivent être arrondis en introduisant, par conséquent, de petites erreurs. Le besoin d'une arithmétique fiable se fait sentir de façon marquée.

Par exemple, en 1982, la bourse de Vancouver a creé un nouvel indice initialisé à la valeur 1000,000. L'indice était recalculé aprés chaque transaction. Vingt-deux mois plus tard, sa valeur était de 524,881. La cause provenait de la valeur de l'indice tronquée après chaque mise à jour au lieu d'être arrondie. Le calcul arrondi aurait donné une valeur de 1098,892. La norme IEEE 754, développée dans les années 80, définit un format de représentation des nombres en virgule flottante ainsi qu'un ensemble d'opérations, de valeurs singulières et d'arrondis permettant de manipuler et d'effectuer des calculs en minimisant les erreurs, mais elle ne les supprime pas.

A la même époque, l'arithmétique par intervalles commence a être réellement reconnue. Le principe de base de cette arithmétique consiste à remplacer une valeur donnée par un intervalle de nombres flottants l'encadrant. Les résultats des calculs,

FIGURE 2.1 – Codage d'un nombre à virgule flottante ; s est le signe, e correspond à l'exposant et m la mantisse

à l'inverse de l'arithmétique en virgule flottante, sont garantis : le résultat exact d'une opération se trouve dans l'intervalle. Ainsi on peut manipuler, sur ordinateur, des nombres qui ne sont pas exactement représentables comme $\frac{1}{3}$ que l'on pourra

encadrer $[0,3333; 0,3334]$. Tout calcul effectué avec cet intervalle inclura dans son résultat la valeur exacte du calcul avec $\frac{1}{3}$.
Ce chapitre détaille, dans une première partie, l'arithmétique en virgule flottante et plus particulièrement la norme IEEE 754. Nous présentons ensuite l'arithmétique par intervalles, ses propriétés algébriques et enfin la notion de surestimation.

2.2 Arithmétique en virgule flottante

D'aprés [53], [52], [1] pour représenter les nombres réels, l'informatique utilise souvent les nombres à virgule flottante ou nombres flottants. Ils sont codés sous la forme d'un signe s, d'un exposant e et d'une mantisse m comme dans la figure 2.2.

Un nombre en virgule flottante n peut donc être représenté par deux nombres m et e (et un signe s) tels que $n = b^e \times m$. Par exemple, le nombre $321,456$ en base 10 deviendra $3,21456 \times 10^2$. Ce format est communément appelé notation scientifique.

2.2.1 La norme IEEE 754

La norme IEEE 754 [53], [52], [1] est un standard définissant la représentation des nombres réels en binaire. C'est le standard le plus couramment utilisé par les ordinateurs pour effectuer des calculs avec des nombres flottants. Cette norme définit un format de représentation des nombres à virgule flottante, ainsi qu'un ensemble d'opérations, de valeurs singulières (par exemple zéro et l'infini) et d'arrondis permettant de manipuler et d'effectuer des calculs sur des nombres.

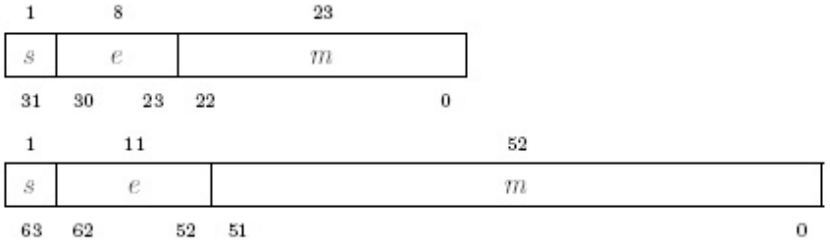

FIGURE 2.2 – Détail du nombre de bits d'un nombre flottant en précision simple et double dans la norme IEEE 754.

2.2.1.1 Description

La norme IEEE 754 décrit deux formats principaux de représentation des nombres réels, la précision simple codée sur 32 bits et la précision double codée sur 64 bits (cf.figure 2.2). Elle définit aussi une précision étendue, respectivement d'au moins 43 bits et 79 bits, pour chacune des deux représentations standard.

Ainsi, l'équation de conversion d'un nombre flottant binaire vers un nombre réel est, sauf cas particulier :

$$(-1)^s \times 2^{e-biais} \times 1.m \tag{2.1}$$

C'est la version binaire de la notation scientifique.

Normalisation

En notation scientifique, la mantisse est normalisée : elle sera toujours supérieure ou égale à 1 et strictement inférieure à 10. Dans la version binaire, on procède de la même façon. La mantisse normalisée sera donc supérieure ou égale à 1 et strictement inférieure à 2. Toutefois, dans certains cas(cf.figure 2.2), une version dénormalisée est utilisée, où la mantisse sera comprise entre 0 et 1. La valeur de l'exposant détermine si le nombre est normalisé ou dénormalisé, le 0 ou le 1 n'est donc jamais stocké.

Biais

Le biais qui apparait dans la partie exposant de l'équation permet de représenter un exposant positif ou négatif en le codant par un nombre toujours positif. Il est fonction du format utilisé et est calculé par l'équation $2^{n-1} - 1$ avec n le nombre de bits de l'exposant. Par exemple, en précision simple, le biais sera de $2^{8-1} - 1 = 127$. Ainsi pour un exposant non biaisé de -100, on stockera, dans le nombre flottant, $e = -100 + 127 = 27$.

Exposant	**Mantisse**	**Valeur**	**qualificatif**
$e = 2^n - 1$	$m \neq 0$	N a N	Not-a-number
$e = 2^n - 1$	$m = 0$	$(-1)^s \infty$	Infini
$0 \prec e \prec 2^n - 1$		$(-1)^s \times 2^{e-biais} \times 1.m$	Normalisé
$e = 0$	$m \neq 0$	$(-1)^s \times 2^{e-biais} \times 0.m$	Dénormalisé
$e = 0$	$m = 0$	$(-1)^s \times 0$	Zéro

FIGURE 2.3 – Les différentes interprétations du codage d'un nombre flottant.[Boldo, 2004]

0	10000001	01011000000000000000000

FIGURE 2.4 – Codage IEEE 754 du nombre 4,625

Signe

Le signe détermine si le nombre est positif ou négatif.

2.2.1.2 Exemple

Comme exemple, convertissons le nombre $4,625$ en un flottant en précision simple. La première étape consiste à le convertir en un binaire à virgule fixe. Le nombre avant la virgule est représenté par des puissances de 2 positives, tandis que le nombre à droite utilise les puissances de 2 négatives.

$$5,425 \left\{ \begin{array}{c} 5 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 101 \\ 0,425 = 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 0,011 \end{array} \right\} 101,011.$$

Puis le résultat est normalisé en décalant la virgule de 2 bits sur la gauche ; l'exposant est biaisé avec la valeur 127 propre aux nombres à virgule flottante en précision simple.

$$101,011 = 1,01011 \times 2^2 \Rightarrow 1,01011 \times 2^{2+127}$$

Soit en binaire

$$1,01011 \times 2^{10000001}$$

Le nombre étant positif, en notation scientifique binaire, on obtient la conversion suivante

$$(-1)^0 \times 2^{10000001} \times 1,01011$$

le codage machine étant illustré par la figure 2.4

2.2.2 Arrondis

Coder une infinité de nombres réels avec un nombre fini de bits est impossible. La plupart des calculs utilisant les nombres réels produisent donc un résultat qui ne

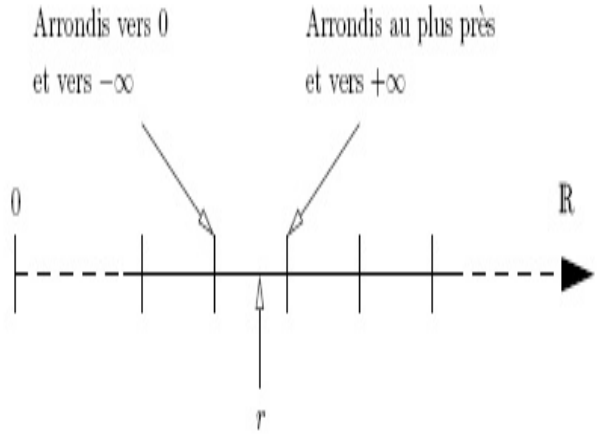

FIGURE 2.5 – Arrondis IEEE 754 d'un nombre réel

peut pas être représenté exactement par un nombre à virgule flottante. Ainsi ces résultats devront souvent être arrondis afin de les faire tenir dans un nombre restreint de bits.
Pour cela, la norme IEEE 754 définit quatre modes d'arrondis illustrés par l'exemple

de la figure 2.4 : au plus proche (arrondi par défaut), vers zéro, vers $+\infty$ et vers $-\infty$.
Ces arrondis sont sources d'erreurs dans les programmes. Pour reprendre un exemple très souvent cité dans la littérature, durant la guerre du golfe en 1991, un anti-missile Patriot a manqué l'interception d'un Scud. Ce dernier a atteint sa cible et a tué vingt-huit soldats américains. L'erreur était due à une accumulation d'erreurs d'arrondis. D'autres exemples montrent le besoin croissant d'une arithmétique fiable et robuste permettant d'effectuer des calculs sans risque d'erreurs, et plus généralement de logiciels fiable. Par exemple, le 30 octobre 1994, le professeur Thomas Nicely de l'université de Lynchburg découvre un bogue dans l'unité de calcul en virgule flottante du Pentium ; il se rendit compte que certaines opérations de division renvoyaient toujours une valeur erronée par excès. Ces erreurs dans la division furent rapidement confirmées par d'autres personnes. L'erreur provenait de l'initialisation incomplète (dans le processeur) d'une table de valeurs servant à un nouvel algorithme de division, plus rapide.

2.3 Arithmétique par intervalles

L'arithmétique par intervalles est une arithmétique définie sur les intervalles plutôt que sur des nombres. Les premières ébauches apparaissent dans les années 1920 [54], [55], [56] mais l'arithmétique par intervalles connaît un réel développement suite à la thèse de Moore en 1962 [57] qui l'a définie de façon très complète et publiée ensuite dans [58]. Une introduction détaillée de l'arithmétique par intervalles est disponible dans [1], [52].

2.3.1 Définitions et notations

Le principe de base de l'arithmétique par intervalles est de représenter une valeur donnée par un intervalle qui l'encadre [1], [53], et d'offrir, à l'inverse de l'arithmétique en virgule flottante, des résultats garantis : en effet le résultat exact d'un calcul est inclus dans l'intervalle retourné.
Un intervalle $\mathbf{x} = [\underline{x}; \overline{x}]$ est défini comme l'ensemble des nombres réels compris entre deux nombres flottants $\underline{x}$ et $\overline{x}$.

$$\mathbf{x} \quad = [\underline{x}; \overline{x}] \quad = \{r \in \Re | \underline{x} \leq r \leq \overline{x}\} \tag{2.2}$$

On désigne par $\Re$ l'ensemble des intervalles bornés fermés sur $\mathbb{R}$. Et à l'instar de $\mathbb{R}^n$, $\Re^n$ est l'ensemble des vecteurs d'intervalles, appelés plus familièrement «boite» ou «pavé», comme illustré par la figure 2.5. Les extrémités de ces vecteurs d'intervalles sont des points dans un ensemble parallélépédique de points aux côtés parallèles aux axes du repère.

2.3.2 Opérations

Il est possible de définir les opérations standard pour les intervalles [1], [53], [59]. Soient $\mathbf{x}$ et $\mathbf{y}$, deux intervalles, et $\diamond \in \{+, -, \times, \div\}$, un opérateur binaire

$$\mathbf{x} \diamond \mathbf{y} = \{x \diamond y | x \in \mathbf{x}, y \in \mathbf{y}\} \tag{2.3}$$

alors

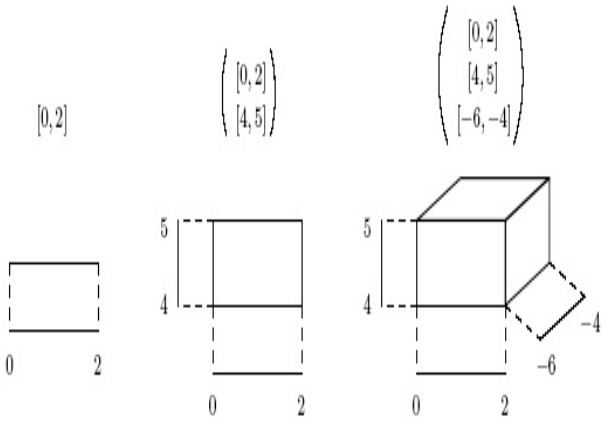

FIGURE 2.6 – Vecteurs d'intervalles dans $\Re$, $\Re^2$ et $\Re^3$ [1]

$$
\begin{aligned}
&[\underline{x};\overline{x}] + [\underline{y};\overline{y}] = \left\lfloor \underline{x} + \underline{y}; \overline{x} + \overline{y} \right\rceil \\
&[\underline{x};\overline{x}] - [\underline{y};\overline{y}] = \left\lfloor \underline{x} - \overline{y}; \overline{x} - \underline{y} \right\rceil \\
&[\underline{x};\overline{x}] \times [\underline{y};\overline{y}] = \left\lfloor min\left(\underline{x}\times\underline{y}, \underline{x}\times\overline{y}, \overline{x}\times\underline{y}, \overline{x}\times\overline{y}\right), max\left(\underline{x}\times\underline{y}, \underline{x}\times\overline{y}, \overline{x}\times\underline{y}, \overline{x}\times\overline{y}\right) \right\rceil \\
&[\underline{x};\overline{x}] \div [\underline{y};\overline{y}] = \left\lfloor min\left(\underline{x}\div\underline{y}, \underline{x}\div\overline{y}, \overline{x}\div\underline{y}, \overline{x}\div\overline{y}\right), max\left(\underline{x}\div\underline{y}, \underline{x}\div\overline{y}, \overline{x}\div\underline{y}, \overline{x}\div\overline{y}\right) \right\rceil
\end{aligned}
\tag{2.4}
$$

avec $0 \notin y$
Ceci se généralise aux opérations algébriques et aux fonctions élémentaires telles que la racine carrée et l'exponentielle.

$$
\sqrt{\mathbf{x}} = \left\lfloor \sqrt{\underline{x}}; \sqrt{\overline{x}} \right\rceil, \qquad \forall x \in \mathbf{x}, x \geq 0
\tag{2.5}
$$

$$
\exp(\mathbf{x}) = \left\lfloor \exp(\underline{x}); exp(\overline{x}) \right\rceil, \quad \forall x \in \mathbf{x}
\tag{2.6}
$$

Dans le cas d'une fonction non monotone comme le cosinus, illustré par la figure 2.6, les intervalles sont décomposés en domaines où la fonction est monotone.

La notation $\lfloor a; b \rceil$ est l'intervalle $\lfloor \alpha; \beta \rceil$, où α est un nombre flottant inférieur ou égal à a, β est un nombre flottant supérieur ou égal à b. Idéalement, il n'y a pas de nombres entre α et a, ni entre β et b : l'intervalle flottant est l'encadrement le plus court possible. Ceci suppose qu'un arrondi exact (il faut comprendre optimal) soit disponible. Longtemps, l'arrondi exact n'a été disponible que pour les opérations arithmétiques simples $\{+, -, \times, \div\}$ et la racine carreé. Récemment, l'équipe

de l'ENS de Jean-Michel Muller à Lyon a proposé des algorithmes d'arrondi exact pour les opérations transcendantes.

Pour garantir les calculs, les bornes inférieures et supérieures sont arrondies respectivement vers $-\infty$ et $+\infty$, tâche réalisée à l'aide de la norme IEEE 754 ;

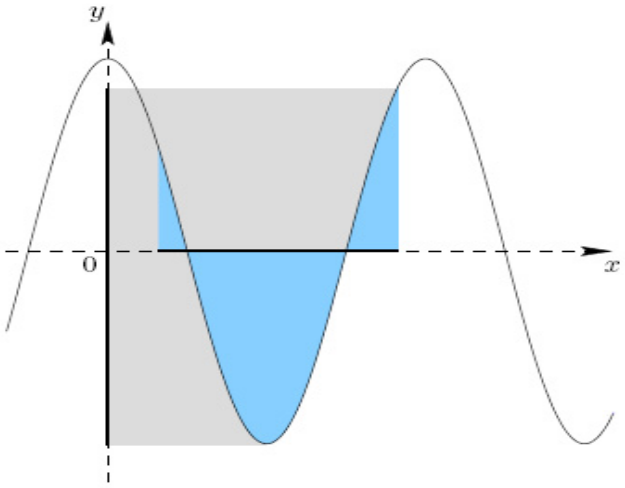

FIGURE 2.7 – Image de l'intervalle d'un cosinus

A l'aide des opérations et fonctions définies ci-dessus, l'arithmétique par intervalles peut être étendue à l'évaluation de n'importe quelle expression mettant en jeu ces calculs. Ainsi pour une fonction f de $D \subset \Re^m$ dans $\Re^n$, la propriété de garantie des résultats de l'arithmétique par intervalles assure que pour tout pavé $\mathbf{x} \subset D$, le pavé obtenu en substituant les variables de f par les intervalles correspondants de $\mathbf{x}$ est un encadrement de l'image de $\mathbf{x}$ par f. Par exemple, si

$$f(x) = x \times (1 - x) \tag{2.7}$$

alors, en remplaçant x par l'intervalle $[0; 1]$, l'intervalle

$$\begin{aligned} f[0;1] &= [0;1] \times (1 - [0;1]) \\ &= [0;1] \times [0;1] \\ &= [0;1] \end{aligned}$$

contient bien l'image exacte $\left[0; \frac{1}{4}\right]$ de l'équation 2.6. On remarque que l'encadrement est loin de l'optimal. Cette surestimation n'est pas due aux arrondis, mais au fait que l'arithmétique par intervalles perd la dépendance entre les variables. Ici on obtient le même résultat que $x \times y$ avec $x, y \in [0; 1]$.

2.3.3 Propriétés algébriques

On peut noter que le passage de l'arithmétique réelle à l'arithmétique par intervalles ne conserve pas les propriétés algébriques [53] , [52], [1].

Ainsi, l'arithmétique par intervalles n'est que sous-distributive, c'est à dire que la distributivité de la multiplication par rapport à l'addition est perdue :

$$\forall \mathbf{x}, \mathbf{y}, \mathbf{z}, \quad \mathbf{x} \times (\mathbf{y} + \mathbf{z}) \quad \subset \quad \mathbf{x} \times \mathbf{y} + \mathbf{x} \times \mathbf{z} \tag{2.8}$$

Si l'on prend l'équation 2.6 précédente dans sa version développée :

$$\begin{aligned} f(x) &= x \times (1 - x) \\ &= x - x^2 \end{aligned}$$

on a

$$\begin{aligned} f([0;1]) &= [0;1] - [0;1]^2 \\ &= [0;1] - [0;1] \\ &= [-1;1] \end{aligned}$$

Des expressions équivalentes en arithmétique réelle ne le sont donc pas forcément en arithmétique par intervalles.

On remarque également que la soustraction n'est plus l'opposé de l'addition :

$$\begin{aligned} [3;4] + [-2;1] &= [1;5] \\ [1;5] - [-2;1] &= [0;7] \supset [3;4] \\ [0;1] - [0;1] &= [-1;1] \end{aligned} \tag{2.9}$$

Le résultat de l'addition moins le second opérande n'est pas égal à $[3;4]$, mais il le contient. Plus généralement, l'amplitude de la somme ou de la différence de deux intervalles est la somme des amplitudes des intervalles, et même un peu plus pour se prémunir contre les erreurs d'arrondis.

De la même façon, la multiplication par lui même d'un intervalle $x = [-2;3]$ n'est pas équivalente à cet intervalle élevé au carré

$$\begin{aligned} [-2;3]^2 &= [0;9] \\ [-2;3] \times [-2;3] &= [-6;9]\,. \end{aligned}$$

Il est normal que $[-1;1]\,[-1;1] = [-1;1]$ car il peut s'agir de 2 variables ou valeurs différentes, il est possible de définir une élévation au carré, telles que $[0;1]^2 = [0;1]$

2.3.4 Surestimation

Ces exemples révèlent la difficulté majeure que présente l'arithmétique par intervalles [53] , [52], [1] : la surestimation (i.e ; un encadrement trop large) des résultats. Elle est due principalement à deux causes : la décorrélation des variables et l'effet enveloppant.

2.3.4.1 Décorrélation des variables

Le premier problème, la décorrélation des variables ou la perte de la dépendance des variables, a été observé dans les exemples de la partie précédente (2.3.3) [1], [53]. Par exemple pour la sous-distributivité, dans un cas on détermine

$$\{x \times (y+z) \,|x \in \mathbf{x}, y \in \mathbf{y}, z \in \mathbf{z}\} \tag{2.10}$$

Tandis que dans le second on calcule

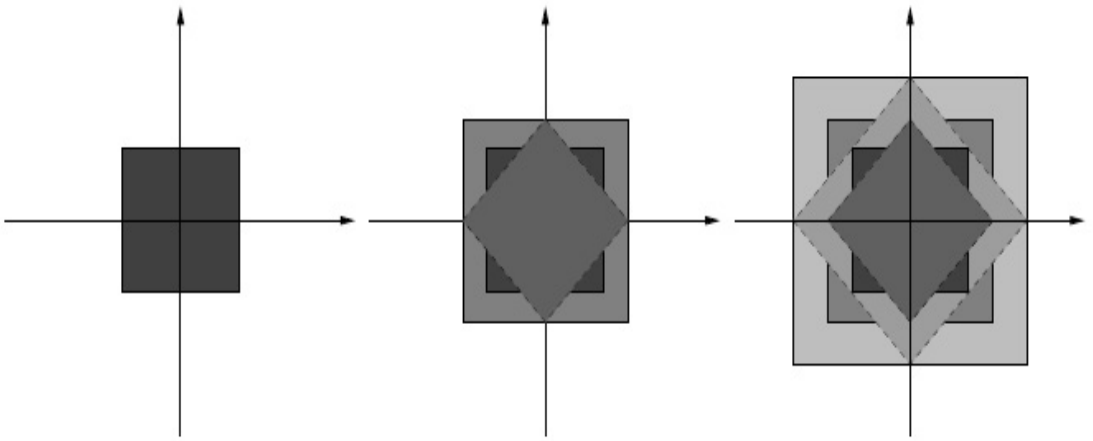

FIGURE 2.8 – Effet enveloppant : après deux rotations successives de $\frac{\pi}{4}$ du petit carré central, on ne retrouve pas le petit carré mais le grand carré [1]. Pis : il en est de même après une rotation de $\frac{\pi}{4}$, puis une rotation de $-\frac{\pi}{4}$

$$\left\{x \times y + x^{'} \times z | x \in \mathbf{x}, x^{'} \in \mathbf{x}, y \in \mathbf{y}, z \in \mathbf{z}\right\} \tag{2.11}$$

De même pour le calcul de $x \times x$, le remplacement de la variable scalaire x par l'intervalle $\mathbf{x}$ donne le résultat

$$\left\{x \times x^{'} | x \in \mathbf{x}, x^{'} \in \mathbf{x},\right\} \tag{2.12}$$

Dans les deux exemples, l'identité entre x et $x^{'}$ est perdue. Il en résulte un intervalle plus large que dans les équations où la variable x n'apparait qu'une seule fois et ne peut donc pas être décorrélée de ses autres occurrences.

2.3.4.2 Effet enveloppant

Le second phénomène, [53] , [52], [1] intrinsèque à l'arithmétique par intervalles, appelé effet enveloppant, ou *wrapping effect*, est dû au fait que l'image exacte d'un pavé $\mathbf{x}$ par une fonction f n'est pas un pavé [60]. Or l'arithmétique par intervalles impose que le résultat calculé soit un vecteur d'intervalles (partie 2.3.1). Ce pavé peut avoir un volume bien plus grand que celui de $\mathbf{x}$ comme l'illustre la figure 2.7

Une première solution pour remédier à ce problème réside en la subdivision du pavé $\mathbf{x}$ en sous-pavés plus petits et à prendre pour solution de $f(\mathbf{x})$ l'union de l'image de ces sous-pavés. Une deuxième solution, proposée par [60], utilise des ellipsoïdes à la place des pavés ; cela résout certes élégamment le problème dans le cas des transformations géométriques affines (i.e. rotations, translations, affinités), mais le problème reste entier pour les transformations non linéaires ; la solution de [60] ne s'applique que dans des situations trés particulières

2.3.5 Variante : Arithmétique par intervalles centrés

Une amélioration de l'arithmétique par intervalles est l'utilisation du calcul centré [53] , [52], [1].

2.3.5.1 Définition 2.1

Soit f une fonction de $\Re^n$ dans $\Re$ continue et différentiable n fois dans $\mathbf{x}$, et $x \in \mathbf{x}$, la formulation centrée de l'arithmétique par intervalles est donnée par.

$$f(\mathbf{x}) \subset f(x) + f'(x)(\mathbf{x}-x) + \frac{f''(x)}{2!}(\mathbf{x}-x)^2 + ... + \frac{f^n(x)}{n!}(\mathbf{x}-x)^{(n)} \tag{2.13}$$

Dans la pratique, on se limitera à la première dérivée, soit

$$f(\mathbf{x}) \subset f(x) + f'(\mathbf{x})(\mathbf{x}-x) \tag{2.14}$$

où $f'(\mathbf{x})$ est évaluée avec l'arithmétique par intervalles naïve ou récursivement. Cette formulation est quasiment minimale lorsque la largeur de $\mathbf{x}$ tend vers zéro.

Le choix de $x \in \mathbf{x}$ est arbitraire. [61] a montré qu'il est possible de maximiser la borne gauche ou de minimiser la borne droite du résultat, ou encore de minimiser la largeur de l'évaluation. Prendre le milieu de $\mathbf{x}$ pour x revient à minimiser l'évaluation, et sera donc notre choix de point central.

Reprenons l'équation 2.6 avec le centre de l'intervalle $\mathbf{x} = [0;1]$ comme point central :

$$\begin{aligned} f(x) &= x \times (1-x) \\ \acute{f}\,(x) &= 1 - 2 \times x \end{aligned}$$

On a

$$\begin{aligned} f(\frac{1}{2}) &= \left(\tfrac{1}{4}\right) \\ \acute{f}\,([0;1]) &= 1 - 2 \times [0;1] = [-1;1] \end{aligned} \tag{2.15}$$

donc

$$f(\mathbf{x}) \in f\left(\frac{1}{2}\right) + \acute{f}\,([0;1]) \times \left([0;1] - \frac{1}{2}\right) = \left[-\frac{1}{4}; \frac{3}{4}\right].$$

Dans ce cas particulier, l'évaluation n'est pas vraiment meilleure que la méthode naïve, mais ce genre de cas est exceptionnel et l'évaluation d'une fonction par la méthode centrée est souvent plus précise.

2.4 Conclusion

L'arithmétique par intervalles, comme son nom le suggère, fait intervenir des intervalles et non plus les types standard. L'idée est d'une part de garantir les résultats en calculant un intervalle dans lequel se trouve le résultat effectif. D'autres parts, on cherche un encadrement précis. On cherche à borner les erreurs d'arrondi dues principalement à l'arithmétique flottante, à la manipulation des données de grande taille et à un nombre important de calculs effectués sur ces données.
L'arithmétique par intervalle a été introduite pour fournir une alternative aux arithmétiques existantes. En effet, seule l'arithmétique exacte répond à un besoin de fiabilité mais elle est lente et ne permet pas d'évaluer les fonctions mathématiques (sin, exp,...) même si elle permet de les manipuler.
Cette arithmétique permet de manipuler des volumes relativement importants de données mais constitue surtout une arithmétique dont l'axe central est de garantir la fiabilité des résultats quand bien même ceux-ci ne s'avèrent pas précis. Le principal point faible de l'arithmétique par intervalles est cet encadrement que l'on obtient de la solution. En effet on peut obtenir un intervalle beaucoup trop large qui entraine une grande imprécision sur le résultat. Les calculs doivent donc être menés de telle sorte que l'encadrement obtenu soit de largeur raisonnable.

Chapitre 3

Calcul et tracé d'une courbe algébrique

3.1 Introduction

Le tracé des surfaces implicites se révèle être une tâche difficile. Les modeleurs géométriques de la CAO utilisent systématiquement des courbes et des surfaces paramétrées ou de subdivision, et non implicites. Dans ce chapitre, un état de l'art des études qui ont été faites sera abordé [59], [62], [63], [64] : l'utilisation de l'arithmétique d'intervalles afin de calculer et de tracer des courbes algébriques ; différentes méthodes basées sur cette arithmétique sont ainsi proposées. La partie 3.2 présente les principes de l'arithmétique d'intervalles et la méthode de subdivision classique de tracé de courbe algébrique ; d'autres méthodes courantes sont également présentées dans cette partie. La partie 3.3 présente [62], [59], [11] une méthode utilisant la base de Bernstein et mettant en oeuvre l'algorithme de De Casteljau, et appelée arithmétique en base de Bernstein : nous prouvons visuellement que le tracé des courbes algébriques par cette méthode est meilleur qu'avec les précédentes, mais ne permet pas de tracer des courbes ayant des fonctions non algébriques (transcendantes) comme le sinus, le cosinus, l'exponentielle, etc .

3.2 Les arithmétiques de calcul d'une fonction algébrique

Cette partie présente une étude qui a été faite par [59] et qui a pour but de présenter au lecteur certaines améliorations apportées sur l'arithmétique des intervalles dans le but de tracer des courbes algébriques, plus particulièrement sur la méthode proposée par Taubin [63], [64] et sur l'arithmétique affine proposée par Comba et Stolfi [65].

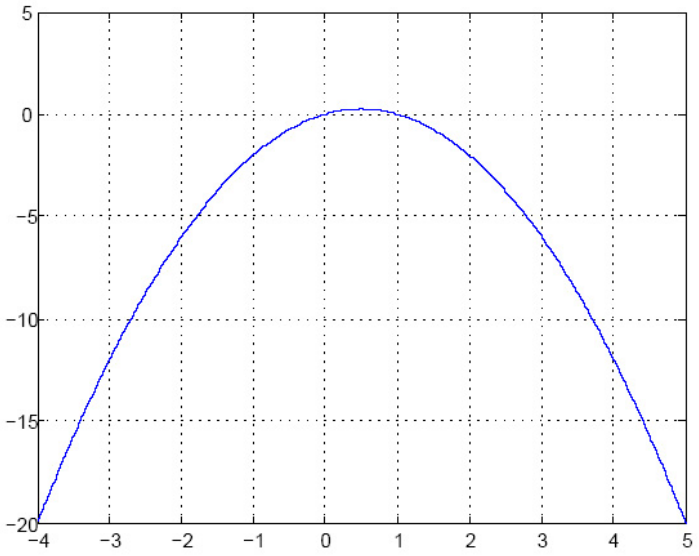

FIGURE 3.1 – Tracé de la fonction $f(x) = x(1-x)$.

3.2.1 L'arithmétique d'intervalles classique

Prenons l'exemple simple de la fonction $f(x) = x\,(1-x)$ tracée dans la figure 3.1, et calculons l'image de l'intervalle $[0,2]$ par f : puisque $x \in [0,2]$ et $(1-x) \in [-1,1]$, on trouve finalement que $f([0,2]) = [-2,2]$. Comme nous pouvons le remarquer, la solution trouvée n'est pas optimale, puisque la solution exacte de l'image de l'intervalle $[0,2]$ par f est l'intervalle $[-2,0.25]$.

Ainsi bien qu'efficace dans le cadre de notre travail, l'arithmétique d'intervalle souffre de quelques inconvénients. Prenons l'exemple de la différence entre deux intervalles, le résultat apparemment évident de l'opération $X - X$ n'est pas 0 : l'arithmétique d'intervalle perd les dépendances entre les variables. La conséquence de cette propriété est l'inflation des intervalles avec le nombre d'opérations effectuées, phénomène aussi connu sous le nom de l'effet enveloppant [66], [60].

Afin d'être totalement fiables, les calculs de borne inférieure sont arrondis vers $-\infty$, tandis que les calculs de borne supérieure sont arrondis vers $+\infty$. Le mode d'approximation est difficile à contrôler dans la pratique, plus particulièrement avec les langages de programmation de haut niveau ; cette précaution n'est en général pas prise en compte, et n'a que peu de conséquence puisque la largeur de l'intervalle est très grande comparée à la différence existant entre deux nombres flottants contigus, et compte tenu des conséquences de cet effet enveloppant.

Afin de tracer la courbe algébrique $f(x,y) = 0$, une méthode de subdivision évalue $f(x,y)$ pour $x \in X = [x^-, x^+]$, $y \in Y = [y^-, y^+]$, ce qui donne l'intervalle $F = [F^-, F^+]$. Si F ne contient pas 0, la courbe n'est pas dans la boite $X \times Y$,

sinon la boite est subdivisée en quatre parties égales, en considérant les milieux des intervalles X et Y. Cette opération est répétée récursivement jusqu'a un certain seuil défini par l'utilisateur ; les boîtes restantes contenant la courbe sont tracées. Puisque l'arithmétique d'intervalles est fiable, aucune boîte n'est oubliée dans la couverture de la courbe. Réciproquement, l'arithmétique d'intervalles étant inflationniste, certaines des boîtes tracées ne sont pas coupées par la courbe. Ce genre de phénomène se vérifie le plus souvent prés des points singuliers de la courbe, ou lorsque la courbure est plus forte (cf.Figure 3.1).

Par la suite, nous nommerons cette méthode comme étant une arithmétique d'intervalles «naïve». Elle suffit pour les calculs opérés sur les polynômes des fonctions algébriques. Cette méthode naïve peut s'étendre à des fonctions non algébriques, telles que les fonctions trigonométriques (sinus, cosinus,tangente, etc...), le logarithme ou encore l'exponentielle.

3.2.2 La forme de Taubin

Une étude de Taubin a montré que l'efficacité de l'arithmétique d'intervalles pouvait être améliorée en exprimant le polynôme $f(x, y)$ ainsi [63], [64]

$$f(x, y) = f_0 + f_1(x, y) + f_2(x, y) + ...f_n(x, y) \tag{3.1}$$

où les $f_i(x, y)$ sont des polynômes homogènes de degré i. Soit F_i la somme des valeurs absolues des coéfficients d'un polynôme homogène f_i alors :

$$f([-a, a], [-a, a]) \in [f_0 - \sum_{i=1}^{n} F_i a^i, f_0 + \sum_{i=1}^{n} F_i a^i] \tag{3.2}$$

Cette méthode de Taubin permet de déterminer que pour $f(x) = x\,(1 - x)$, $f([0, 1])$ se situe dans $[0, 0.25]$, intervalle plus fin que $[0, 1]$ fourni par l'arithmétique d'intervalles naïve vue précédemment. Taubin utilise cette expression basée sur les polynômes homogènes pour le tracé de courbes algébriques 2D $f(x, y) = 0$ et la couverture ("voxellisation") de surfaces algébriques 3D $f(x, y, z) = 0$.

En utilisant la même technique de subdivision que précédemment. la méthode de Taubin se révèle bien plus efficace car elle permet d'obtenir des résultats plus précis pour le tracé de courbes algébriques : en effet la surestimation est beaucoup moins importante si la fonction algébrique est exprimée sous la forme de Taubin. La figure 3.2 illustre parfaitement ce résultat avec l'exemple de l'ovale de Cassini sur l'intervalle $[-2, 2] \times [-2, 2]$: le tracé autour de l'origine, point double de la courbe, est mieux approché et l'espace comporte moins de subdivisions inutiles qu'avec l'arithmétique naîve.

3.2.3 L'arithmétique affine

L'arithmétique affine a été proposée par Comba et Stolfi dans [65], et plus tard par Figueiredo et al. [67], pour diminuer l'inflation importante de l'arithmétique

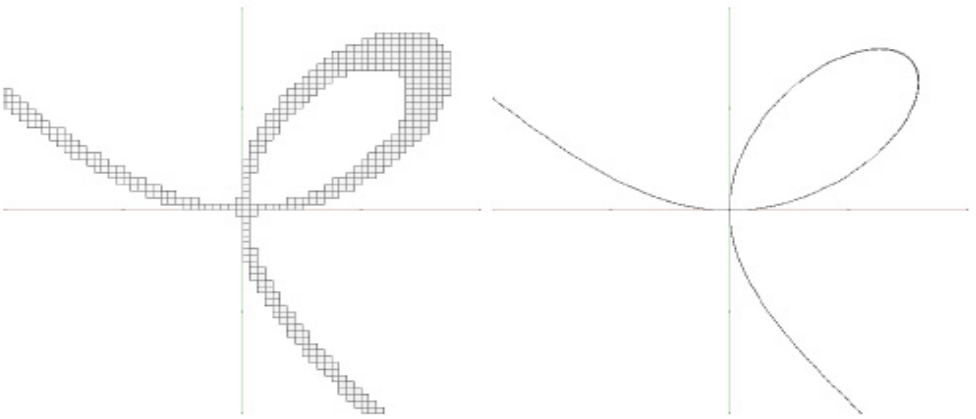

FIGURE 3.2 – L'arithmétique d'intervalles est fiable, mais inflationniste : illustration sur le folium de Descartes, pour $(x, y) \in [-2, 2] \times [-2, 2]$. A gauche, le folium est couvert par un ensemble de boites, utilisant l'arithmétique d'intervalles et la méthode de subdivision, à droite, un tracé plus précis de la courbe réelle

d'intervalles classique. La taille des intervalles est diminuée de telle sorte que les corrélations de premier ordre entre les variables sont prises en compte ; une quantité x est représentée par sa forme affine :

$$\hat{x} = x_0 + x_1\epsilon_1 + x_2\epsilon_2 + ...x_n\epsilon_n \tag{3.3}$$

où les x_i sont des nombres réels et les ϵ_i sont des réels inconnus et indépendants compris dans un intervalle $[-1, 1]$. x_0 est la valeur «centrale» de $\hat{x}$ et les autres x_i sont ses dérivées partielles, tandis que les ϵ_i sont des variables de «bruits». Chacune de ces variables représente une composante indépendante de l'incertitude totale de la quantité idéale associée.

Comme précédemment, les opérations standard peuvent être définies avec l'arithmétique affine. Etant donnée une valeur x dans l'intervalle $[a, b]$, la forme affine correspondante de x est $\hat{x} = x_0 + x_1\epsilon_1$ où $x_0 = (b + a)/2$ et $x_1 = (b - a)/2$; une même variable de bruit peut apparaître dans deux voire plusieurs quantités, montrant ainsi leurs dépendances ; ces variables sont utilisées pour empêcher les extrémités des intervalles des expressions de grandir aussi vite qu'elles le feraient avec l'arithmétique d'intervalles classique. Cette méthode requiert de bonnes approximations sous forme affine pour les expressions de haut degré et pour les variables de bruits appropriées. Un calcul d'encadrement est plus coûteux en temps que l'arithmétique d'intervalles standard mais plus précis, donc moins de subdivisions sont à calculer.

Appliquée au tracé de courbes algébriques, cette méthode obtient des résultats bien meilleurs que ceux de l'arithmétique d'intervalles naïve, et sensiblement meilleurs que la solution proposée par Taubin (cf. comparaison de la figure 3.2 avec

les différentes figures). L'algorithme de subdivision employé avec cette arithmétique affine ne subdivise pas simultanement l'espace suivant les deux dimensions x et y, mais successivement, d'où un découpage de l'espace différent des précédents sur la figure 3.2, appliqué sur la méthode naïve pour une meilleure comparaison visuelle.

3.2.4 Premier bilan des méthodes présentées

Après avoir introduit l'arithmétique d'intervalles dite naïve pour le tracé des courbes algébriques, puis cité deux études contribuant à l'amélioration de l'évaluation de fonctions polynômiales par intervalles (méthode de Taubin et arithmétique affine), un premier constat est permis : aucune de ces méthodes ne présente une subdivision optimale de l'espace, qui permettrait une approximation satisfaisante du tracé de courbe algébrique.

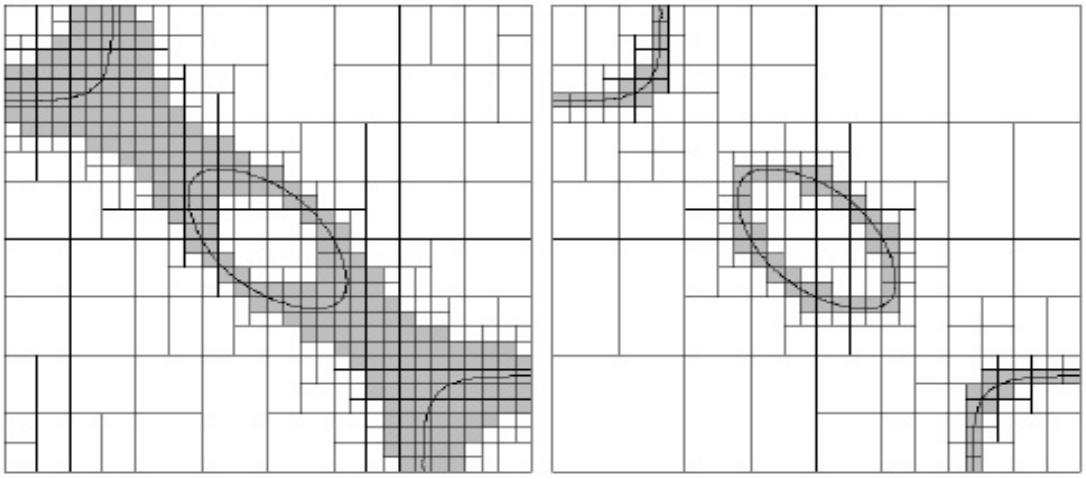

FIGURE 3.3 – Comparaison entre deux méthodes. A gauche, l'arithmétique d'intervalles naïve ; à droite, l'arithmétique affine de Figueiredo et al.Courbe $f(x, y) = x^2 + y^2 + xy - (xy)^2/2 - 1/4, x \in [-2, 2] \times [-2, 2]$

Martin et al ; dans [68], produisent une étude plus étendue sur les différentes techniques de tracé jusque là existantes ; néanmoins, si certaines d'entres elles se montrent plus efficaces que d'autres, aucune n'offre une approximation plus précise que les méthodes présentées auparavant dans ce chapitre ; cette étude propose également l'utilisation de la base de Bernstein comme moyen d'approcher l'image d'un intervalle par une fonction algébrique, mais les résultats présentés ne sont pas meilleurs ; Martin et al ; n'ont fait qu'exprimer le polynôme dans la base de Bernstein : nous proposons dans la partie suivante une méthode reprenant cette idée, mais en ajoutant à la méthode de subdivision l'algorithme classique de De Casteljau, qui permet de déterminer un ensemble de points de contrôle de la courbe plus fin à partir du précédent, donnant comme nous allons le voir une subdivision de l'espace quasi-optimale lors du tracé, et donc une bien meilleure approximation de la courbe.

3.3 La méthode de Bernstein - de Casteljau

Les propriétés de la base de Bernstein [62], [59], [11] sont bien connues des communautés de la CAO et l'informatique graphique, depuis les travaux de Bernstein, Bézier et de De Casteljau. Cette base permet d'avoir une estimation précise des intervalles pour les polynômes [69], [70], [71]. D'autres communautés ont également travaillé récemment sur la base de Bernstein [72], [73], [74] et un solveur en base de Bernstein pour les systèmes polynomiaux, décrit par Mourrain et Pavone, est disponible dans GALAD, voir [75]. Cette partie donne quelques définitions et rappels sur la base de Bernstein et ses principales propriétes, et présente l'algorithme de de Casteljau comme une amélioration de la méthode de Bernstein pour tracer des courbes et surfaces algébriques.

3.3.1 Convention dans la base de Bernstein

3.3.1.1 Principe de base

Les fonctions polynômiales sont communément écrites dans la base canonique, c'est à dire pour un polynôme de degré n en x

$$f(x) = a_0 + a_1 x + a_2 x + \ldots, a_n x^n \tag{3.4}$$

ou encore, sous la forme matricielle :

$$f(X) = XF \tag{3.5}$$

avec $X = (1, x, x^2, ...x^n)$ pour variables et $F = (a_0, a_1, a_2, \ldots, a_n)^T$ pour coefficients. Les fonctions algébriques des systèmes de contraintes géométriques dont nous disposons sont données sous cette forme ; il faut être donc capable de les convertir dans la base de Bernstein exprimée par :

$$B = (B_{0,n}(x), B_{1,n}(x), \ldots, B_{n,n}(x)) \tag{3.6}$$

avec $B_{i,n}(x) = \binom{n}{i} x^i (1-x)^{n-i}$ et $x \in [0, 1]$. Cette opération de conversion entre la base canonique et la base de Bernstein est une application linéaire, représentable par une matrice M de taille $(n+1) \times (n+1)$ telle que $B = XM$. La forme générale de M pour un polynôme de degré n est donnée par :

$$M = \begin{pmatrix} m_{00} & 0 & \cdots & 0 \\ m_{10} & m_{11} & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ m_{n0} & m_{n1} & \ldots & m_{nn} \end{pmatrix} \text{ avec } m_{ij} = \binom{n}{j}\binom{n-j}{i-j}(-1)^{i-j}. \tag{3.7}$$

En d'autres termes, la base canonique et la base de Bernstein sont liées par les relations :

$$x^i = \sum_{j=i}^{n} \frac{\binom{j}{i}}{\binom{n}{i}} B_{j,n}(x) \tag{3.8}$$

et

$$B_{i,n}(x) = \sum_{j=i}^{n} (-1)^{j-i} \binom{n}{j} \binom{j}{i} x^j \tag{3.9}$$

3.3.2 Application aux courbes algébriques

Dans le cas des courbes algébriques, [11] le degré n de la fonction f est fixé par l'utilisateur. De par la formule générale d'une courbe algébrique, notre polynôme est de degré au plus m en x et n en y, on peut écrire la forme matricielle de f dans la base canonique.

$$f(x, y) = XFY^T = 0, \tag{3.10}$$

avec $X = (1, x, x^2, ..., x^m)$ et $Y = (1, y, y^2, ...y^n)$. En appliquant le changement de base et en rappelant que $B = XM$, on obtient :

$$\begin{aligned} f(x, y) &= X.F.Y^T \\ &= (B_x M^{-1}).F.(B_y M^{-1})^T \\ &= B_x (M^{-1} F (M^{-1})^T).B_y^T \\ &= B_x.B.B_y^T \end{aligned} \tag{3.11}$$

où la matrice B contient les points de contrôle de la courbe,
$B_x = (B_{0,m}(x), B_{1,m}(x), \ldots, B_{n,m}(x))$ est le vecteur de la base de Bernstein en x et $B_y = (B_{0,n}(y), B_{1,n}(y), \ldots, B_{n,n}(y))$ le vecteur de la base de Bernstein en y. Puisque nous nous intéressons aux fonctions algébriques et que nous devons considérer les coefficients du polynôme comme les inconnues du problème, le degré maximal des monômes en x et en y est identique, donc ici $m = n$.
Une particularité des polynômes de Bernstein est de s'appliquer sur un intervalle restreint à $[0, 1]$ pour chaque variable x et y définies respectivement sur des intervalles $[x_0, x_1]$ et $[y_0, y_1]$. On crée ainsi les nouvelles variables $x^{'}$ et $y^{'}$ telles que :

$$\left\{ \begin{array}{c} x = (x_1 - x_0)x^{'} + x_0 = ax^{'} + b, \text{avec } x \in [x_0, x_1] \text{et } x^{'} \in [0, 1] \\ y = (y_1 - y_0)y^{'} + y_0 = cy^{'} + d, \text{avec } y \in [y_0, y_1] \text{et } y^{'} \in [0, 1] \end{array} \right\} \tag{3.12}$$

d'où la matrice de changement de variable pour X :

$$X = X^{'}U_x = (1, x^{'}, x^{'2}, \ldots, x^{'n}) \begin{pmatrix} u_{x_{00}} & u_{x_{01}} & \cdots & u_{x_{0n}} \\ 0 & u_{x_{10}} & \ddots & u_{x_{1n}} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & u_{x_{nn}} \end{pmatrix} \tag{3.13}$$

Avec $u_{x_{i,j}} = \binom{i}{j} a^i b^{j-i}$ et a et b définis par le changement de variable de l'Equation 3.14. Le changement de variable pour Y se fait de manière similaire. En intégrant ceci au calcul de changement de base, l'Equation 3.13 devient :

$$\begin{aligned} f(x,y) &= X.F.Y^T \\ &= (X^{'}U_x).F.(Y^{'}U_y)^T \qquad (\textit{changement de variables}) \\ &= X^{'}.(U_x.F.U_y^T)Y^{'T} \\ &= (B_{x^{'}}.M^{-1}).(U_x.F.U_y^T)(B_{y^{'}}M^{-1})^T \quad (\textit{changement de base}) \\ &= B_{x^{'}}.(M^{-1}.U_x.F.U_y^T(M^{-1})^T).B_{y^{'}}^T \\ &= B_{x^{'}}.B.B_{y^{'}}^T \end{aligned} \tag{3.14}$$

où comme précédemment la matrice B contient les points de contrôle de la courbe, $B_{x^{'}}$ est le vecteur de la base de Bernstein en $x^{'}$ et $B_{y^{'}}$ le vecteur de la base de Bernstein en $y^{'}$.

3.3.3 Exemple d'une courbe algébrique : le folium de Descartes (première partie) [59]

Cet exemple a pour but d'illustrer sur un cas simple la conversion d'une fonction algébrique de la base canonique vers la base de Bernstein. Nous reprenons l'exemple du folium de Descartes déjà vue auparavant, qui est une courbe algébrique de degré 3 et d'équations :

$$f(x,y) = x^3 + y^3 - 3xy = 0 \tag{3.15}$$

Graphiquement, il s'agit d'une boucle dont le noeud se situe à l'origine du repère cartésien (cf.Figure 3.1). Nous allons traduire dans cette partie son équation de la base canonique à la base de Bernstein en utilisant les outils données précédemment. Sous la forme matricielle, le follium peut s'écrire :

$$f(x,y) = XFY^T = (1, x, x^2, x^3) \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ y \\ y^2 \\ y^3 \end{pmatrix} \tag{3.16}$$

Et la matrice de conversion pour un polynôme de degré 3 est donnée par :

$$M = \begin{pmatrix} 1 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix}, \ d'ou \ \ M^{-1} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & \frac{1}{3} & 0 & 0 \\ 1 & \frac{2}{3} & \frac{1}{3} & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \tag{3.17}$$

D'aprés l'Equation 3.12, $f(x,y) = B_x.B.B_y^T$, avec $B = M^{-1}.F.(M^{-1})^T$. On trouve

le résultat suivant pour la matrice des points de contrôle B de f :

$$B = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & -\frac{1}{3} & -\frac{2}{3} & 0 \\ 0 & -\frac{2}{3} & -\frac{4}{3} & -1 \\ 1 & 0 & -1 & -1 \end{pmatrix} \tag{3.18}$$

En développant ce résultat, on trouve l'équation de f dans la base de Bernstein :

$$\begin{aligned} f(x,y) = B_{3,3}(x).B_{0,3}(y) - \frac{1}{3}B_{1,3}(x).B_{1,3}(y) - \frac{2}{3}B_{2,3}(x).B_{1,3}(y) \\ - \frac{2}{3}B_{1,3}(x).B_{2,3}(y) - \frac{4}{3}B_{2,3}(x).B_{2,3}(y) - B_{3,3}(x).B_{2,3}(y) \\ + B_{0,3}(x).B_{3,3}(y) - B_{2,3}(x).B_{3,3}(y) - B_{3,3}(x).B_{3,3}(y) \end{aligned}$$

Dans la partie suivante, cet exemple sera repris afin d'illustrer l'algorithme de Casteljau utilisé dans la méthode de subdivision.

3.3.4 L'algorithme de de Casteljau

3.3.4.1 Principe

La méthode de subdivision [62], [59] classique présentée dans les parties précédentes peut être améliorée lorsqu'on utilise la base de Bernstein pour exprimer l'équation de notre courbe algébrique, par l'utilisation de l'algorithme de De Casteljau : grâce à cet algorithme les encadrements des intervalles subdivisés sont déterminés plus rapidement que par la méthode classique.

La courbe algébrique $f(x,y) = 0$ dans le carré $[0,1] \times [0,1]$ peut être considérée comme la courbe d'intersection entre le plan $z = 0$ et le carreau de Bézier défini par $x = x$, $y = y$ et $z = f(x,y)$. Dans le cas général si f est de degré m en x et de degré n en y, les points de contrôle du carreau de Bézier sont $P_{i,j} = (i/m, j/n, z_{i,j})$, avec $i = 0, \ldots, m$ et $j = 0 \ldots, n$. Les coefficients $z_{i,j}$ sont calculés comme montré précédemment : $f(x,y) = XFY^t$ avec $X = (1, x, x^2, \ldots, x^m)$ et $Y = (1, y, y^2, \ldots, y^n)$.

Puisque $f(x,y) = XFY^t = B_x B B_y^t$ dans la base de Bernstein, la matrice B contient les points de contrôle de la surface. La propriété de l'enveloppe convexe dans la base de Bernstein garantit que $f(x,y)$ est dans l'enveloppe convexe définie par ses points de contrôle. Ici pour $z = f(x,y)$ avec $x \in [0,1]$ et $y \in [0,1]$, l'enveloppe est simplement l'intervalle $[min(z_{i,j}), max(z_{i,j})]$.

Si cet intervalle contient 0, l'algorithme classique de de casteljau permet de diviser le carreau en deux ou quatre sous-carreaux [62]. Cette procédure évite toute conversion dans la base canonique. Numériquement, la matrice des points de contrôle B est divisée par l'algorithme de De Casteljau pour obtenir de nouveaux points de contrôle. En appliquant ce procédé récursivement sur chaque nouveau carreau on obtient une meilleure approximation de la courbe. La Figure 3.3 reprend à titre de

comparaison l'exemple de l'ovale de cassini vu .

L'algorithme de Casteljau subdivise la matrice B des points de contrôle suivant la dimension x de l'espace, sachant que les lignes de B correspondent à la dimension x et les colonnes à la dimension y. Ensuite, il suffit de répéter cet algorithme suivant la dimension y sur chacune des deux sous-matrices obtenues, afin d'obtenir les quatre sous-matrices de la subdivision. La partie suivante donne un exemple simple d'application de l'algorithme.

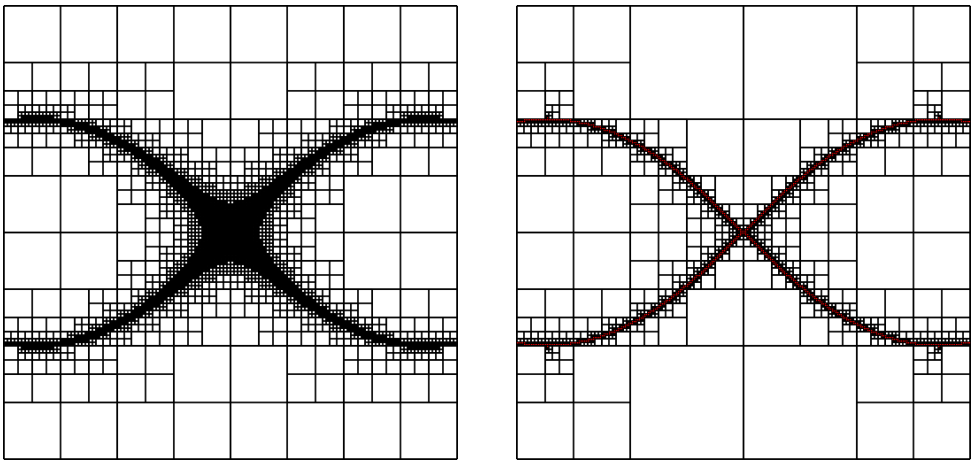

FIGURE 3.4 – Comparaison entre deux méthodes. A gauche,L'arithmétique d'intervalles naïve ; à droite, la méthode de Bernstein-De Casteljau, sur la courbe de l'ovale de Cassini avec huit niveaux de subdivision

3.3.4.2 Exemple d'une courbe algébrique : Le folium de Descartes (deuxième partie) [59]

Reprenons maintenant l'exemple du folium de Descartes, dont nous avons calculé la matrice de points de contrôle B pour $(x, y) \in [0, 1] \times [0, 1]$ dans le paragraphe 3.3.3. Nous allons voir comment dérouler l'algorithme de De Casteljau sur B suivant la dimension x, pour obtenir les deux sous-matrices B_x^1 et B_x^2 définies respectivement dans $[0, \frac{1}{2}] \times [0, 1]$ et dans $[\frac{1}{2}, 1] \times [0, 1]$.
Intéressons-nous dans le premier temps aux éléments de la première colonne de B :

$$B = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & -\frac{1}{3} & -\frac{2}{3} & 0 \\ 0 & -\frac{2}{3} & -\frac{4}{3} & -1 \\ 1 & 0 & -1 & -1 \end{pmatrix}. \tag{3.19}$$

En lui appliquant l'algorithme de De Casteljau successivement avec le coefficient $\frac{1}{2}$, on obtient la première colonne de B_x^1 et la première colonne de B_x^2 :

$$B_x^1 = \begin{pmatrix} 0 & . & . & . \\ 0 & . & . & . \\ 0 & . & . & . \\ \frac{1}{8} & . & . & . \end{pmatrix} \quad et \quad B_x^2 = \begin{pmatrix} \frac{1}{8} & . & . & . \\ \frac{1}{4} & . & . & . \\ \frac{1}{2} & . & . & . \\ 1 & . & . & . \end{pmatrix} \tag{3.20}$$

La même opération est effectuée sur les trois autres colonnes de B pour trouver l'intégralité des matrices B_x^1 et B_x^2 et on trouve finalement que :

$$B_x^1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & -\frac{1}{6} & -\frac{1}{3} & \frac{1}{2} \\ 0 & -\frac{1}{4} & -\frac{2}{3} & 0 \\ \frac{1}{8} & -\frac{1}{3} & -\frac{7}{8} & -\frac{3}{8} \end{pmatrix} \quad et \quad B_x^2 = \begin{pmatrix} \frac{1}{8} & -\frac{1}{3} & -\frac{7}{8} & -\frac{3}{8} \\ \frac{1}{4} & -\frac{5}{12} & -\frac{13}{12} & -\frac{3}{4} \\ \frac{1}{2} & -\frac{1}{3} & -\frac{7}{6} & -1 \\ 1 & 0 & -1 & -1 \end{pmatrix} \tag{3.21}$$

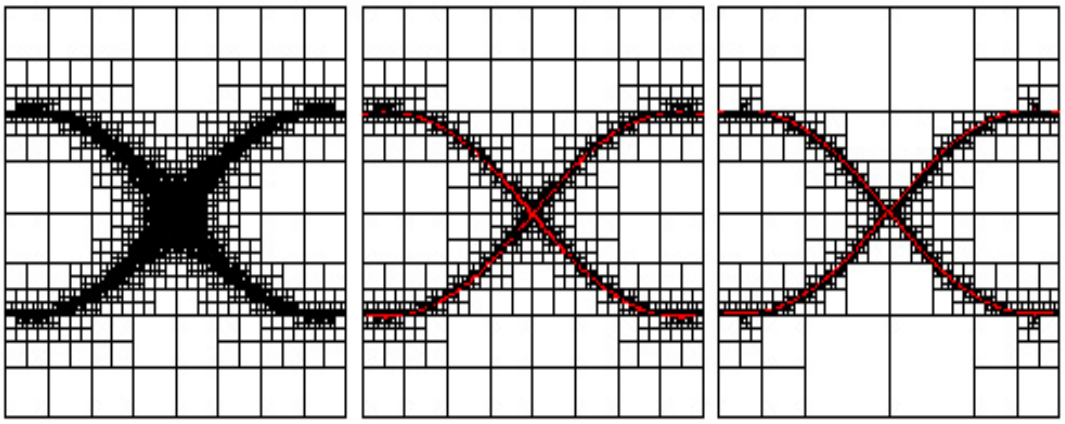

FIGURE 3.5 – Comparaison entre trois méthodes. A gauche, l'arithmétique d'intervalles naïve ; au milieu méthode de Taubin, à droite algorithme de Casteljau sur la courbe de l'ovale de Cassini avec huit subdivisions

Pour obtenir les quatre sous-matrices correspondant à une subdivision de l'intervalle de départ en quatre, il suffit d'appliquer l'algorithme de De Casteljau sur chacune des matrices B_x^1 et B_x^2 suivant la dimension y, c'est à dire en subdivisant les lignes de ces matrices. On obtient de la sorte les quatre sous-matrices B_{xy}^{11} , B_{xy}^{12}, B_{xy}^{21} et B_{xy}^{22}.

En répétant l'opération jusqu'a un certain seuil, on trouve finalement un ensemble de sous-matrices correspondant chacune à une zone de l'espace, et qui permet d'approcher la courbe algébrique définie par la matrice départ B.

3.3.5 Conclusion sur la méthode de Bernstein-De Casteljau

L'utilisation de la base de Bernstein pour l'expression de la fonction algébrique et de l'algorithme de De Casteljau dans la méthode de subdivision donne lieu à des résultats trés intéressants. En effet, on constate une nette amélioration visuelle du tracé de la courbe, en comparaison des méthodes évoquées auparavant ; la subdivision de l'espace est optimale dans tous les cas puisqu'aucun intervalle n'est subdivisé

inutilement lors du calcul. [62], [59], [11]

Martin et al [68] comparent différentes méthodes de tracé sur une courbe de degré 4 : la Figure 3.5 donne les résultats comparatifs pour cette courbe avec respectivement les méthodes d'arithmétique d'intervalles nave, de Taubin et de Bernstein-De Casteljau sur l'intervalle $[0,1] \times [0,1]$. En ne considérant que le tracé de la courbe sur cet exemple, les deux dernières méthodes semblent donner des résultats à peu prés équivalents ; en observant plus attentivement la manière avec laquelle ont été obtenus ces résultats, autrement dit les subdivisions appliquées sur l'intervalle de départ, on constate clairement que la dernière méthode est bien plus efficace : la forme de Taubin implique un nombre élevé de subdivisions superflues contrairement à la forme de Bernstein associée à l'algorithme de de Casteljau ; cet écart entre les différentes techniques de tracé semblent s'accroître en augmentant le degré de la courbe comme le montre la Figure 3.6.

Malgré des résultats convaincants, la méthode de bernstein présente tout de même quelques limitations :

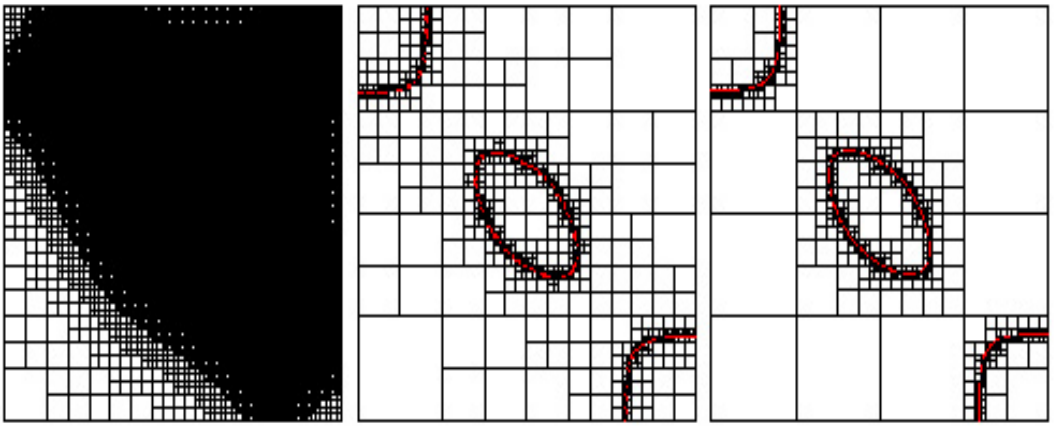

FIGURE 3.6 – La courbe de Martin et al., avec les méthodes d'arithmétiques d'intervalles naïve, de Taubin et de Bernstein

- Pour les degrés élevés (20 au plus), le conditionnement de la matrice de conversion de la base canonique vers la base de Bernstein devient mauvais.
- La méthode de Bernstein ne peut s'appliquer sur les fonctions dites transcendantes telle le sinus, l'exponentielle, etc. De telles fonctions peuvent être bornées par deux polynômes : cette méthode est utilisée dans l'arithmétique des ordinateurs (calculs en virgule flottante) pour le calcul de fonctions transcendantes ; d'autres possibilités reposent sur la base de Poisson [76], ou la forme de Taylor-Bernstein [73], [74].

.

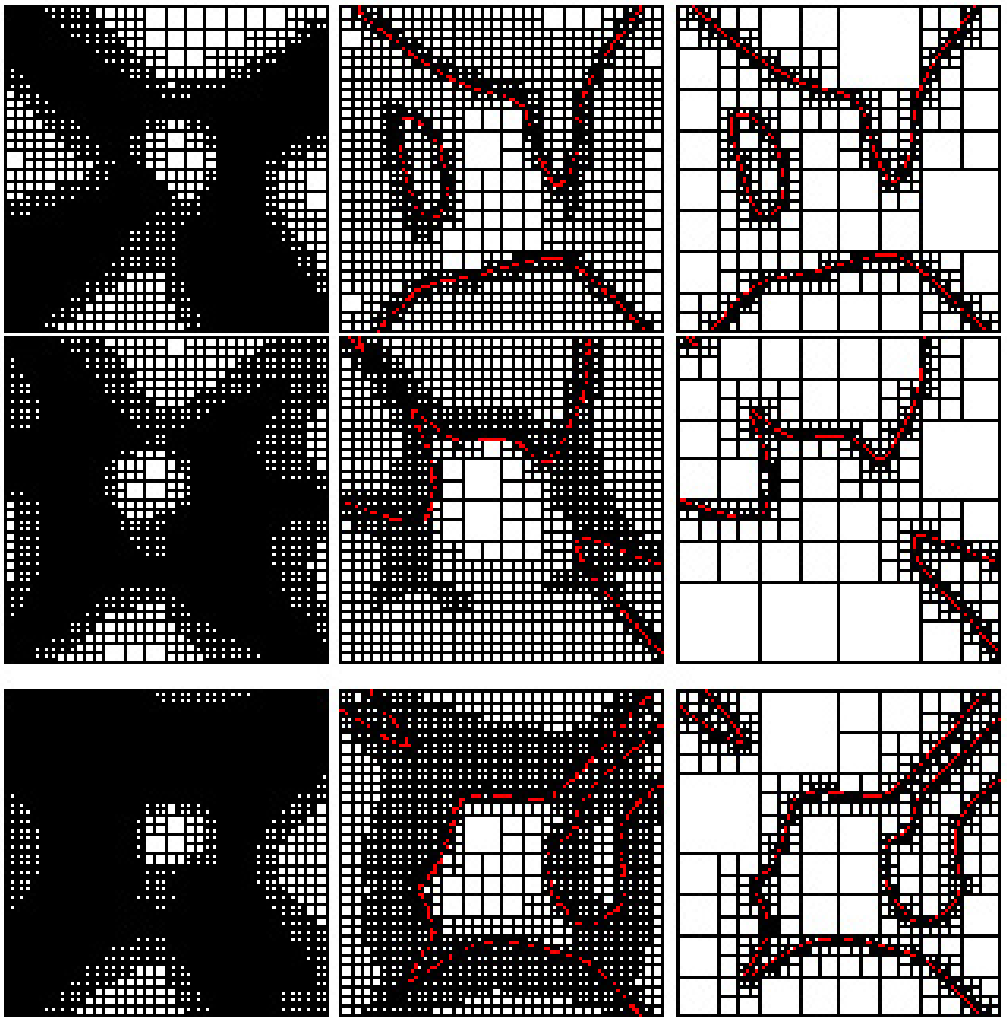

FIGURE 3.7 – Courbes aléatoires de degrés respectifs 10, 14 et 18. A gauche, arithmétique d'intervalles naïve ; au centre, forme de Taubin ; à droite, méthode de Bernstein-De Casteljau

3.4 Conclusion

Pour les équations de faible degré, les différentes méthodes citées donnent en général des résultats à peu près équivalents. Il est évident que l'arithmétique naïve est optimale lorsque chaque variable n'apparait qu'une seule fois dans l'expression à évaluer ; mais dés que le degré ou que le nombre de monômes augmente, ces méthodes ont des comportements bien différents. Les méthodes de Taubin et d'arithmétique affine sont clairement plus efficaces que l'arithmétique naïve, et la méthode de Bernstein est bien meilleure que celle de Taubin. Martin et al [68] ne sont pas arrivés aux mêmes conclusions, probablement parce qu'ils n'ont considéré que des polynômes de faible degré ou avec peu de monômes. De plus, ils n'utilisent pas l'algorithme de de

Casteljau en combinaison avec la base de Bernstein. Dans tous les cas, la méthode de Bernstein-de Casteljau donne une subdivision quasi-optimale de l'espace, et est donc meilleure que les autres.

Chapitre 4

Solveur et bases tensorielles de Bernstein

4.1 Le principe

Cette partie présente les solveurs classiques utilisant les bases de Bernstein.

Soit $f(x) = 0$, avec $x = (x_1, \dots x_n)$ et $f = (f_1, \dots f_n)$ le système à résoudre. Les solveurs actuels utilisant les bases de Bernstein trouvent toutes les racines réelles contenues dans un pavé donné $[u, v]$ de $\mathbb{R}^n$. Leur principe est le même que celui des solveurs de type Newton par intervalles : le pavé initial est empilé ; tant que la pile de pavés est non vide, le pavé en sommet de pile est dépilé et étudié.

Pour étudier un pavé $p = [u, v] = \{x | u_i \leq x_i \leq v_i\}$, le solveur commence par calculer un pavé $p' = [u', v'] \subset [u, v]$ inclus dans p et qui contient les mêmes racines que p. Le paragraphe §4.4.4 présente la méthode utilisée pour la réduction de p en p'. Après la réduction, trois cas sont possibles :

- soit p' est vide, et alors p ne contient aucune racine ;
- soit p' est trop petit pour être encore subdivisé ; alors, p' contient une racine simple (le jacobien est de rang n dans p'), ou plusieurs racines simples proches, ou une racine multiple : des tests permettant de prouver l'existence et l'unicité peuvent éventuellement être utilisés ; ils sont donnés en §4.5 ; p', après un éventuel étiquetage (contient-il une racine simple ? le jacobien s'annule-t-il dans p' ? etc), est inséré dans une liste des solutions du système ; certains solveurs permettent de préciser la racine simple contenu dans un pavé en appliquant quelques itérations de Newton à partir du centre du pavé ;
- soit le pavé est encore gros, mais n'est plus significativement réduit ; il contient probablement plusieurs racines ; il est coupé en deux, le long de son côté le plus long, ou bien le long du côté qui a été le moins réduit, et les deux sous pavés résultant de la subdivision sont empilés. La subdivision est le seul moyen de séparer plusieurs racines distinctes ; mais elle a un coût exponentiel ; l'idéal est de ne l'utiliser que pour séparer les racines.

Les solveurs utilisant les propriétés des bases tensorielles de Bernstein (ou les solveurs de Bernstein par concision) ressemblent beaucoup aux solveurs de type Newton par intervalles. Cependant, les solveurs de type Newton par intervalles, comme celui

de ALIAS [77], utilisent une autre réduction que celle, spécifique aux solveurs de Bernstein, qui est présentée en §4.4.4. Pour réduire le pavé p, les solveurs de type Newton par intervalles itèrent $p \leftarrow p \cap n(p)$ où $n(p)$ est un encadrement de n : pour $x \in p$. $n(x) = x - Mf(x)$ est l'itération de Newton, et M est une inverse approximative du jacobien du polynôme f au centre du pavé p.

Les solveurs de type Newton par intervalles utilisent typiquement une arithmétique d'intervalles centrée pour calculer l'encadrement de $n(x \in p)$: pour calculer l'encadrement d'un polynôme $f(x \in p)$ sur un pavé p, ils utilisent : $f(x \in p) \subset f(c) + (p - c)f'(p)$, où c est le centre du pavé p, et $f'(p)$ est un encadrement de $f'(x \in p)$ calculé avec une arithmétique d'intervalles naïve. Les bases tensorielles de Bernstein fournissent de meilleurs encadrements que l'arithmétique d'intervalles, et de meilleures réductions. Cela est illustré par les figures 4.5, et 5.6.

Le reste de cette partie est structuré ainsi : le paragraphe §4.2 rappelle les définitions et les propriétés essentielles des bases tensorielles de Bernstein, présente la conversion de la base canonique à la base de Bernstein, et montre l'utilisation de ces bases pour la conversion et l'encadrement des polynômes univariés et multivariés. Le paragraphe §4.3 donne les formulations matricielles pour la méthode de Casteljau et pour les conversions entre bases. Le paragraphe §4.4 présente quelques algorithmes utilisant les bases tensorielles de Bernstein pour le calcul de couvertures de courbes implicites, pour l'isolation des racines réelles de polynômes univariés, pour le calcul de l'enveloppe convexe en 2D, ainsi que pour le préconditionnement et la réduction de pavés. Le paragraphe §4.5 présente quelques tests utiles pour prouver l'existence ou l'absence de racines, et l'unicité de racine dans un pavé. Les principales limitations des solveurs fondées sur les bases tensorielles de Bernstein sont rappelées dans le paragraphe §4.6.

4.2 Définitions et propriétés essentielles

Soit $C(k,d) = \binom{d}{k} = \frac{d!}{(d-k)!\,k!}$ le nombre de sous-ensemble de k éléments dans un ensembles de d éléments ; nous utilisons la convention : $C(k,d) = 0$ quand k ou d est négatif.

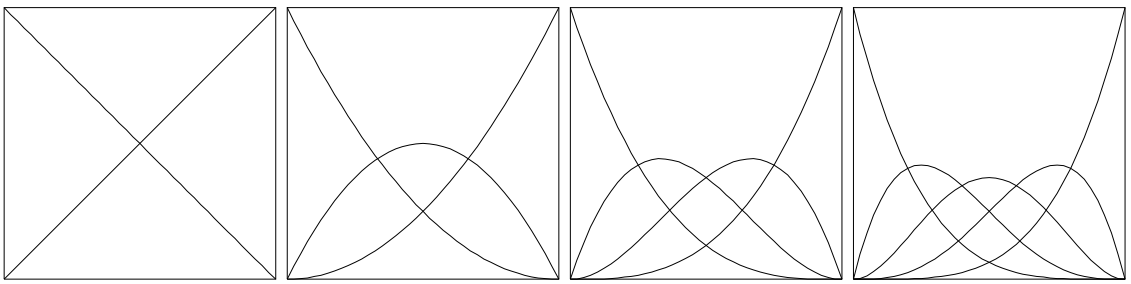

FIGURE 4.1 – Les polynômes de Bernstein pour les degrés 1, 2, 3, 4, de gauche à droite. Chaque carré est le carré unité. Les valeurs maximum et minimum de $B_i^{(d)}(x)$ se produisent en $x = i/d$.

Les $d+1$ polynômes de Bernstein $B_i^{(d)}(x)$, $i = 0, \ldots, d$, $x \in \mathbb{R}$, de degré d (aussi notés $B_i(x)$ quand le degré est sous-entendu) constituent une base des polynômes de degré d. Par définition :

$$B_i^{(d)}(x) = \binom{d}{i} x^i (1-x)^{d-i}$$

Les graphes $(x,\ B_i^{(d)}(x))$ sont représentés en Figure 4.1 pour $x^i = 1, 2, 3, 4$ et $d = 4$. Ces polynômes apparaissent dans l'expansion de Newton de $(x + (1-x))^d$:

$$(x + (1-x))^d = 1 = \sum_{i=0}^{d} \binom{d}{i} x^i (1-x)^{d-i} = \sum_{i=0}^{d} B_i^{(d)}(x)$$

La conversion entre la base canonique : $(1, x, x^2, \ldots x^d)$ et la base de Bernstein : $(\ B_0^{(d)}(x), \ldots\ B_d^{(d)}(x))$ est une transformation linéaire, représentable par une matrice carrée d'ordre $d+1$. Une formule classique est [78] :

$$x^k = (1/\ C(k,d)) \sum_{i=k}^{d} C(k,i)\ B_i^{(d)}(x)$$

On en déduit, pour $k = 1$:

$$x = (1/d) \times \sum_{i=0}^{d} i\ B_i^{(d)}(x)$$

et pour $k = 0$, on retrouve la formule :

$$x^0 = 1 = \sum_{i=0}^{d} B_i^{(d)}(x)$$

Les propriétés essentielles de la base de Bernstein résultent de leur définition ; la somme $\sum_{i=0}^{d} B_i^{(d)}(x)$ égale 1 ; pour $x \in [0,1]$, ils sont tous positifs ou nuls. Il en résulte que pour $x \in [0,1]$, $p(x) = \sum_i p_i\ B_i^{(d)}(x)$ est une combinaison linéaire convexe des coefficients $p_i, i = 0, \ldots n$.

Pour un polynôme $p(x) \in \mathbb{R}[x]$, chaque coefficient p_i est un nombre réel, et $p(x \in [0,1])$ se trouve dans l'enveloppe convexe des p_i, qui est l'intervalle $[min_i p_i, max_i p_i]$. Cet encadrement est précis, et sa borne inférieure, ou supérieure, est atteinte quand elle se produit en $i = 0$ ou en $i = d$. En effet, $B_0^{(d)}(0) = 1$ et $B_{i>0}^{(d)}(0) = 0 \Rightarrow p(0) = p_0$; d'autre part, $B_d^{(d)}(1) = 1$ et $B_{i<d}^{(d)}(1) = 0 \Rightarrow p(1) = p_d$.

Si les p_i sont des points de $\mathbb{R}^2$, ou de $\mathbb{R}^3$, alors l'ensemble $p(x \in [0,1])$ décrit un arc de courbe, appelé courbe de Bézier ; cet arc est inclus dans l'enveloppe convexe des coefficients p_i, qui sont appelés points de contrôle.

Exemple : comme $x = 0\ B_0(x) + 1/d\ B_1(x) + 2/d\ B_2(x) + \ldots d/d\ B_d(x)$, la courbe polynomiale $(x, y = p(x))$, avec $x \in [0,1]$, est incluse dans l'enveloppe convexe de

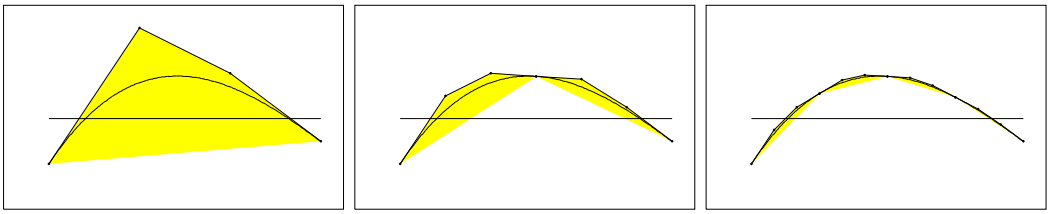

FIGURE 4.2 – Deux itérations de la méthode de Casteljau sur une courbe de Bézier $(x, y = f(x))$, où f est un polynôme de degré 3. Les enveloppes convexes des points de contrôle sont affichées.

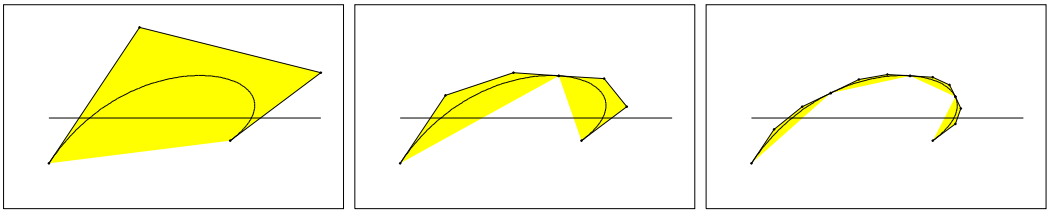

FIGURE 4.3 – Deux itérations de la méthode de Casteljau sur une courbe de Bézier $(x = f_1(u), y = f_2(u))$, $u \in [0, 1]$, de degré 3. Il y a donc 4 points de contrôle $p_i = (x_i, y_i)$. Les enveloppes convexes des points de contrôle sont affichées.

ses points de contrôle $(i/d, p_i)$. On en déduit que la courbe $(x, y = p(x))$ passe par le premier et par le dernier point de contrôle, $(0, p_0)$ et $(1, p_d)$.

Contrairement aux coefficients dans la base canonique usuelle : $(1, x, x^2, \ldots x^d)$, les points de contrôle (i.e., les coefficients dans la base de Bernstein, ou coefficients de Bernstein) de $p(x)$ dépendent de l'intervalle de l'argument x. Un algorithme classique dû à Paul de Casteljau [78] calcule les points de contrôle pour le sous intervalle $x \in [0, t]$ et le sous intervalle $x \in [t, 1]$, sans repasser par la base canonique. La valeur $t = 1/2$ est la plus souvent utilisée.

4.2.1 Méthode de Casteljau

A partir des coefficients de Bernstein d'un polynôme $f(x), x \in [0, 1]$, la méthode de Casteljau calcule les coefficients de Bernstein de sa moitié gauche : $x \in [0, 1/2]$, et de sa moitié droite $x \in [1/2, 1]$. Les Figures 4.2 et 4.3 montrent deux itérations de l'algorithme sur un polynôme $y = f(x)$ et sur une courbe de Bézier. La Figure 4.4 montre l'arbre des calculs de la méthode.

Soit $b^{(d)} = b = (b_0, b_1, \ldots b_d)$ les points de contrôle d'un polynôme univarié $y = f(x)$, $x \in [0, 1]$. L'algorithme calcule le vecteur des milieux : $b^{d-1} = ((b_0 + b_1)/2, \ldots (b_i + b_{i+1}))/2 \ldots)$. Le vecteur $b^{(d-1)}$ a un élément de moins que $b^{(d)}$. Puis l'algorithme calcule de même $b^{(d-2)}$ à partir de $b^{(d-1)}$, etc, jusqu'à obtenir $b^{(0)}$, qui contient un seul élément. En formule : $b_i^{(k)} = (b_i^{(k+1)} + b_{i+1}^{(k+1)})/2$, pour $k = d-1, \ldots 0$ et $i = 0, \ldots k$.

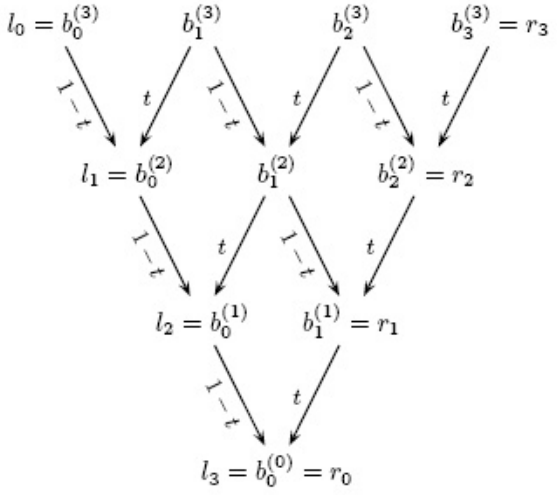

FIGURE 4.4 – $b^{(d)} = (b_i^{(d)})$ est le vecteur des coefficients de Bernstein d'un arc de degré $d = 3$. La Figure montre les interpolations linéaires calculées par la méthode de de Casteljau, pour trouver les vecteurs $l = (l_i)$ et $r = (r_i)$ des coefficients de Bernstein de la moitié gauche et de la moitié droite de l'arc.

Les premiers éléments de $b^{(d)}, b^{(d-1)}, \ldots b^{(0)}$ donnent le vecteur $l = (l_i)$ des coefficients de Bernstein de la moitié gauche $x \in [0, 1/2]$ de $f(x)$, et les derniers éléments de $b^{(d)}, b^{(d-1)}, \ldots b^{(0)}$ donnent le vecteur $r = (r_i)$ des coefficients de Bernstein de la moitié droite $x \in [1/2, 1]$ de $f(x)$. En formule, la moitié gauche de $f(x)$ a pour coefficients de Bernstein : $l_i = b_0^{(d-i)}$ et la moitié droite a pour coefficients de Bernstein : $r_i = b_i^{(i)}$, pour $i = 0, \ldots d$.

Plutôt que des coefficients 1/2,1/2, il est possible d'utiliser des coéfficients $1-t$,t, ce qui fournit les points de contrôle pour le sous intervalle gauche $x \in [0, t]$, et pour le sous intervalle droit $x \in [t, 1]$. Il est également possible d'utiliser une valeur de t hors de $[0, 1]$, mais la stabilité numérique de l'algorithme est perdue ; quand $|t|$ croit, l'instabilité numérique augmente.

Ce calcul $b_i^{(k)} = (b_i^{(k+1)} + b_{i+1}^{(k+1)})/2$ s'applique en toute dimension, quand les $b_i^{(k+1)}, b_{i+1}^{(k+1)}$ sont des points et non plus des valeurs scalaires. Aussi la méthode de de Casteljau s'applique-t-elle en toute dimension.

4.2.2 Conversion entre bases dans le cas univarié

La conversion entre bases peut être effectuée matriciellement, comme en §4.3.2, mais pour le programmeur, la méthode qui suit est plus commode ; par exemple, elle s'étend au cas multivarié (§4.2.4). Il faut d'abord définir une fonction qui calcule la contribution du monôme x^k au i ième polynôme de Bernstein pour le degré d ; c'est

$$W([0, 1], x^k,\ B_i^{(d)}) =\ C(k, i)/\ C(k, d)$$

Jusqu'à maintenant, nous avons considéré que x appartient à l'intervalle $[0,1]$. Pour encadrer les valeurs du polynôme $f(x)$ pour $x \in [u,v]$, il suffit d'utiliser la transformation affine qui applique $[u,v]$ sur $[0,1]$: $x = u + (v-u)x'$. Quand x décrit $[u,v]$, x' décrit $[0,1]$. Donc, en posant $w = v - u$, la contribution de x^k, avec $x \in [u,v]$, à $B_i^{(d)}(x)$, est : $W([u,v], x^k,\ B_i^{(d)}(x)) = W([0,1], (u+(v-u)x)^k,\ B_i^{(d)}(x))$:

$$\begin{aligned} W([u,v], x^k,\ B_i^{(d)}(x)) &= W([0,1], (u+(v-u)x)^k,\ B_i^{(d)}(x)) \\ &= W([0,1], \sum_{j=0}^{k} C(j,k) \times w^j \times x^j \times u^{k-j},\ B_i^{(d)}(x)) \\ &= \sum_{j=0}^{k} C(j,k) \times w^j \times u^{k-j} \times\ W([0,1], x^j,\ B_i^{(d)}(x)) \\ &= \sum_{j=0}^{\min(k,i)} w^j \times u^{k-j} \times\ C(j,k) \times\ C(j,i)/\ C(j,d) \end{aligned}$$

La formule est valable pour tout w, positif, négatif, ou nul. Ensuite, pour convertir de la base canonique vers la base de Bernstein, un polynôme $f(x)$ de degré d dans un pavé $x \in [u,v]$, il faut d'abord initialiser tous les coefficients de Bernstein à 0, puis pour chaque monôme canonique $c_k \times x^k$ du polynôme $f(x)$, et pour tous les i entiers dans $0, 1, \ldots d$, il faut incrémenter de $c_k \times\ W([u,v], x^k,\ B_i^{(d)})$ le coefficient du i^{ime} polynôme de Bernstein $B_i^{(d)}$. Les deux boucles sur x^k et le polynôme de Bernstein $B_i^{(d)}$ peuvent être imbriquées dans n'importe quel ordre ; de même les valeurs pour les indices i et k peuvent être considérées dans n'importe quel ordre. Si les monômes $c_k x^k$ de $f(x)$ sont stockés dans une liste, alors la même puissance peut apparaître plusieurs fois : il n'est pas utile de réduire le polynôme.

4.2.3 Base tensorielle de Bernstein pour les polynômes multivariés

Pour illustrer le cas des polynômes multivariés, et par simplicité, nous considérons des polynômes bivariés en x et y. La base canonique est le produit cartésien de la base canonique en x : $(x^0, x^1, \ldots x^d)$ et de la base canonique en y : $(y^0, y^1, \ldots y^e)$, où d est le degré maximum en x et e le degré maximum en y. d et e peuvent être différents. Pour $d = 3, e = 2$, la base canonique est : (x^0y^0, x^0y^1, x^0y^2, x^1y^0, x^1y^1, x^1y^2, x^2y^0, x^2y^1, x^2y^2, x^3y^0, x^3y^1, x^3y^2).

De même la base tensorielle de Bernstein est le produit de la base de Bernstein en x : ($B_0^{(d)}(x)$, $B_1^{(d)}(x), \ldots$ $B_d^{(d)}(x)$) et de la base de Bernstein en y : ($B_0^{(e)}(y)$, $B_1^{(e)}(y), \ldots$ $B_e^{(e)}(y)$). Les bases tensorielles canoniques et de Bernstein ont donc même cardinalité $(d+1) \times (e+1)$ (d'ailleurs, toutes les bases d'un espace vectoriel ont même cardinalité, en dimension finie). La notation $B_{i,j}^{(d,e)}(x,y) = B_i^{(d)}(x) \times\ B_j^{(e)}(y)$ est utilisée par commodité.

La généralisation à un nombre quelconque de variables est évidente.La propriété dite de l'enveloppe convexe s'étend au cas multivarié : pour les mêmes raisons que dans le cas univarié, les valeurs d'un polynôme multivarié dans un pavé sont comprises entre le plus petit et le plus grand des coefficients de Bernstein. La borne de cet encadrement est atteinte quand elle se produit en un sommet du pavé, *i.e.,* chacun des indices i_k est soit nul soit maximum, autrement dit égal au degré d_k en la k^{ieme} variable.

4.2.4 Conversion tensorielle pour un polynôme multivarié

Usuellement, un polynôme bivarié est exprimé comme un ensemble de termes $c_{k,l}x^k y^l$ dans la base canonique. La contribution de chaque terme $c_{k,l}x^k y^l$ au polynôme de Bernstein $B_{i,j}^{(d,e)}(x,y) = B_i^{(d)}(x) \times B_j^{(e)}(y)$ est $c_{k,l}$ fois le produit de la contribution de x^k à $B_i^{(d)}(x)$ et de la contribution de y^l à $B_j^{(d)}(y)$. En formule :

$$W([0,1], c_{k,l}x^k y^l,\ B_{i,j}^{(d,e)}(x,y)) = \\ c_{k,l} \times W([0,1], x^k,\ B_i^{(d)}(x)) \times W([0,1], y^l,\ B_j^{(e)}(y)) \quad (4.1)$$

Généralisons au cas multivarié. Pour convertir un polynôme en n variables de la base canonique à la base tensorielle de Bernstein, il faut initialiser à zéro tous les coefficients dans la base tensorielle de Bernstein. Puis, pour chaque terme $c_k x^k = c_{k_1,k_2,\ldots k_n} x_1^{k_1} x_2^{k_2} \ldots x_n^{k_n}$ du polynôme, et pour chaque multi-indice $i = i_1 i_2 \ldots i_n$, avec $0 \leq i_j \leq d_j$, et pour $j = 1, \ldots n$, il faut ajouter la contribution du terme $c_k x^k$ au polynôme de Bernstein $B_i^{(d)} = B_{i_1}^{(d_1)}(x_1) \times B_{i_2}^{(d_2)}(x_2) \ldots \times B_{i_n}^{(d_n)}(x_n)$.

Généralisons maintenant l'hypercube $[0,1]^n$ au pavé $[u,v]$, $u = (u_1, u_2, \ldots u_n)$ et $v = (v_1, v_2, \ldots v_n)$. Soit $w = (w_i = v_i - u_i)$. La contribution de chaque terme $c_k x^k$ au polynôme de Bernstein B_i est alors (la formule est correcte pour tout w_i, négatif, positif ou nul) :

$$W([u,v], c_k x^k,\ B_i(x)) = \\ c_{k_1,k_2,\ldots k_n} \times W([u_1,v_1], x_1^{k_1},\ B_{i_1}^{(d_1)}(x_1)) \times \ldots \times W([u_n,v_n], x_n^{k_n},\ B_{i_n}^{(d_n)}(x_n))$$

4.2.5 Encadrement d'un polynôme multivarié sur un pavé

Pour calculer un intervalle englobant les valeurs d'un polynôme multivarié sur un pavé $[u,v]$, où $u = (u_1, u_2, \ldots u_n)$ et $v = (v_1, v_2, \ldots v_n)$, les coefficients du polynôme dans la base tensorielle de Bernstein sont calculés, soit en convertissant de la base canonique à la base tensorielle de Bernstein, soit en utilisant l'algorithme de Paul de Casteljau, quand le pavé est issu de la subdivision d'un pavé plus grand. L'encadrement du polynôme est alors donné par l'intervalle du plus petit coefficient de Bernstein, et du plus grand coefficient de Bernstein du polynôme. Une borne de cet encadrement est atteinte quand elle se produit en un sommet du pavé. Les encadrements calculés ainsi sont bien meilleurs que ceux calculés par l'arithmétique d'intervalles, comme en témoigne la Figure 4.5

4.3 Formulation matricielle

Ce paragraphe donne une formulation matricielle pour la conversion entre les bases, et pour la méthode de subdivision de Casteljau.

4.3.1 Formulation matricielle de la méthode de Casteljau

La matrice $P^{(d)}$, ou P quand le degré d est sous-entendu, parfois appelée matrice de Pascal, est une matrice carrée d'ordre $d+1$, telle que $P^{(d)}_{l,c} = C(l,c)$, en utilisant la convention que les indices de ligne et de colonne commencent par 0. C'est une matrice triangulaire supérieure, qui contient les $d+1$ premières lignes du triangle de Pascal. L'inverse de $P^{(d)}$ est vraiment remarquable. Par exemple, pour $d = 4$, P et P^{-1} sont :

$$P^{(4)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 & 6 \\ 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \qquad (P^{(4)})^{-1} = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 \\ 0 & 1 & -2 & 3 & -4 \\ 0 & 0 & 1 & -3 & 6 \\ 0 & 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$G^{(d)} = P^{(d)} \times \mathrm{diag}(1, 1/2, 1/4, \ldots 1/2^d)$ où G pourrait être appelée la matrice de de Casteljau à gauche. Les colonnes de $G^{(d)}$ sont les colonnes de $P^{(d)}$, divisées par des puissances croissantes de 2. $G^{(d)}$ apparaît dans la méthode de Casteljau : si $b = (b_0, b_1, \ldots b_d)$ est le vecteur des coefficients de Bernstein d'un polynôme univarié, alors b G est le vecteur des coefficients de Bernstein de la moitié gauche du polynôme. De même, $D^{(d)}$ ou D pourrait être appelée la matrice de Casteljau à droite, et b D est le vecteur des coefficients de Bernstein de la moitié droite du polynôme. $D^{(d)}$ a les mêmes colonnes que $G^{(d)}$ mais en ordre inverse. Par exemple, pour le degré $d = 4$:

$$G^{(4)} = \begin{pmatrix} 1 & 1/2 & 1/4 & 1/8 & 1/16 \\ 0 & 1/2 & 2/4 & 3/8 & 4/16 \\ 0 & 0 & 1/4 & 3/8 & 6/16 \\ 0 & 0 & 0 & 1/8 & 4/16 \\ 0 & 0 & 0 & 0 & 1/16 \end{pmatrix}$$

Si, dans la méthode de Casteljau, les coefficients sont $1-t, t$ au lieu de $1/2, 1/2$ (autrement dit si le domaine $[0,1]$ est divisé en $[0,t]$ et $[t,1-t]$), alors la matrice de Casteljau à gauche devient : $G^{(d)} = \mathrm{diag}(1, \beta, \beta^2, \ldots \beta^d)\ P^{(d)} \mathrm{diag}(1, \alpha, \alpha^2, \ldots \alpha^d)$ où $\alpha = 1-t, \beta = t/\alpha$. De nouveau, la matrice droite $D^{(d)}$ a les mêmes colonnes que $G^{(d)}$ mais en ordre inverse.

4.3.2 Conversion matricielle dans le cas univarié

Ce paragraphe donne une méthode matricielle pour le changement de base. La conversion de la base canonique $X = (x^0, x^1, \ldots x^d)$ à la base de Bernstein $B = (B_0(x), B_1(x), \ldots B_d(x))$ est une transformation linéaire représentable par une

matrice M carrée d'ordre $d+1$. Par commodité, nous supposons que les indices de ligne et de colonne de M et M^{-1} commencent en 0. Alors :

$$X = BM, \quad M_{i,k} = W([0,1], x^k, B_i^{(d)}) = C(k,i)/\ C(k,d)$$

Inversement :

$$XM^{-1} = B, \quad (M^{-1})_{l,c} = C(c,d) \times C(l-c, d-c) \times (-1)^{l+c}$$

Le polynôme $f(x) = \sum_{k=0}^{d} c_k x^k$ est représenté dans la base canonique par le vecteur $c = (c_0, c_1, \ldots c_d)$, ou encore : $f(x) = Xc^t$. Sa représentation dans la base de Bernstein est $f(x) = \sum_{i=0}^{d} b_i B_i^{(d)}(x)$, et est stockée dans le vecteur $b = (b_0, b_1, \ldots b_d)$. De plus $X = BM$ où M est la matrice définie dans le paragraphe précédent. De $f(x) = Xc^t = Bb^t$ et $X = BM$, on déduit $b^t = Mc^t$.

Par exemple, pour le degré $d = 4$, les matrices M et M^{-1} sont :

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1/2 & 0 & 0 & 0 \\ 1 & 1/2 & 1/6 & 0 & 0 \\ 1 & 3/4 & 1/2 & 1/4 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \qquad M^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 12 & -12 & 4 & 0 \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix}$$

Le lecteur intrigué par les ressemblances entre M et P pourra vérifier que $M^{-1} = \Theta \times \mathrm{diag}_{k=0}^{d}(\ C(k,d)) \times P^t \times \Theta$ où Θ est la matrice diagonale d'ordre $d+1$: $\Theta = \mathrm{diag}(-1, 1, -1, 1, \ldots)$.

La formulation matricielle met bien en évidence certaines propriétés combinatoires. Aussi, elle est parfois commode, par exemple pour étudier le conditionnement des matrices de conversion entre les deux bases. Classiquement, il résulte d'une telle étude que les flottants en simple précision (sur 32 bits) deviennent insuffisamment précis pour des degrés plus grands ou égaux à 10, et les flottants en double précision (sur 64 bits) pour des degrés plus grands ou égaux à 20.

4.3.3 Conversion matricielle dans le cas bivarié

Posons $X = (x^0, x^1, \ldots x^d)$, $Y = (y^0, y^1, \ldots y^e)$. Le polynôme dans la base canonique est $f(x,y) = XCY^t$, avec C une matrice ayant $d+1$ lignes et $e+1$ colonnes. $C_{r,c}$ est le coefficient de $x^r y^c$. En utilisant $X = B_x M_x$ et $Y = B_y M_y$ avec les conventions naturelles :

$$f(x,y) = XCY^t = B_x M_x C(\ B_y M_y)^t = B_x (M_x C M_y^t)\ B_y^t$$

donc les coefficients de $f(x,y)$ dans la base tensorielle de Bernstein sont donnés par la matrice $T = M_x C M_y^t$, donc $f(x,y) = B_x T\ B_y^t$.

La formulation matricielle précédente est souvent utilisée en CFAO et en informatique graphique pour les surfaces de Bézier : elles sont décrites par $(x = X(u,v), y = Y(u,v), z = Z(u,v))$, où $(u,v) \in [0,1]^2$; les polynômes X, Y, Z sont donnés dans

la base de Bernstein, par un tableau à deux entrées p_{ij} de points 3D : (x_{ij}, y_{ij}, z_{ij}). Malheureusement, ce formalisme matriciel s'étend mal au cas trivarié (les fonctions paramétriques volumiques, parfois nommées "volumes de Bézier") et au delà : les matrices sont des tableaux à 2 dimensions. Il vaut donc mieux utiliser la méthode de conversion tensorielle en §4.2.2.

4.4 Algorithmes utilisant les bases tensorielles de Bernstein

4.4.1 Calcul de couvertures de courbes implicites

Les bases tensorielles de Bernstein sont souvent utilisées pour calculer des couvertures des courbes ou surfaces algébriques implicites données par leur équation : $f(x, y) = 0$, ou $f(x, y, z) = 0$ [12]. Une couverture d'une courbe ou surface est un ensemble discret de pixels, de voxels, etc, qui contient la courbe ou surface en question. Une bonne couverture doit être la moins épaisse possible : idéalement, tous ses pixels, voxels, etc ont une intersection non vide avec la courbe ou la surface couverte.

La Figure 4.5 montre des couvertures de courbes implicites polynomiales $f(x, y) = 0$ calculées avec une méthode de subdivision récursive du carré initial : un intervalle encadrant $f(x, y)$ pour $(x, y) \in P$ est calculé ; si l'intervalle ne contient pas 0, alors le pavé P ne peut contenir de points de la courbe d'équation $f(x, y) = 0$; sinon, si le pavé est suffisamment petit, il est inséré dans la couverture de la courbe, ou affiché ; sinon, le pavé est divisé en 4 pavés de côté 2 fois plus petit, qui sont étudiés à leur tour, récursivement, ou en utilisant une pile ou une autre structure d'attente.

L'intervalle encadrant les valeurs de f dans un pavé donné P peut être calculé par une arithmétique d'intervalles, naïve ou centrée. C'est ce qui est fait dans la rangée du haut de la Figure 4.5. L'encadrement peut aussi être calculé en utilisant les bases tensorielles de Bernstein, comme en §4.2.5. Pour cela, le polynôme est converti dans la base de Bernstein ; l'encadrement est donné par le coefficient de Bernstein le plus petit, et par le plus grand. Au besoin, la méthode de Casteljau est utilisée pour calculer les coefficients de Bernstein de f dans les quatre pavés les plus petits ; un premier appel de la méthode de Paul de Casteljau calcule les coefficients de la moitié gauche et de la moitié droite ; puis, pour la moitié gauche, un second appel calcule les coefficients du quart inférieur gauche à partir des coefficients de la moitié gauche ; un troisième appel calcule les coefficients du quart inférieur droit à partir des coefficients de la moitié droite. La Figure 4.5 montre les couvertures obtenues par cette méthode dans la rangée inférieure. Clairement, la méthode utilisant les bases de Bernstein écarte beaucoup plus rapidement les pavés non traversés par la courbe d'équation $f(x, y) = 0$. Cette méthode est aussi utilisée en 3D pour calculer des couvertures de surfaces algébriques données par leur équation implicite : $f(x, y, z) = 0$, ou des couvertures de courbes 3D décrites par un système $f(x, y, z) = g(x, y, z) = 0$.

Cette méthode de calcul de couverture de courbe continue en 2D a été généralisée

à des courbes fractales définies par des IFS (*Iterated Functions Systems*, systèmes de fonctions itérées), et plus récemment à des attracteurs étranges [79,80]. Ceci suggère que les solveurs présentés dans cette thèse peuvent être généralisés à des systèmes d'équations représentant des objets non lisses.

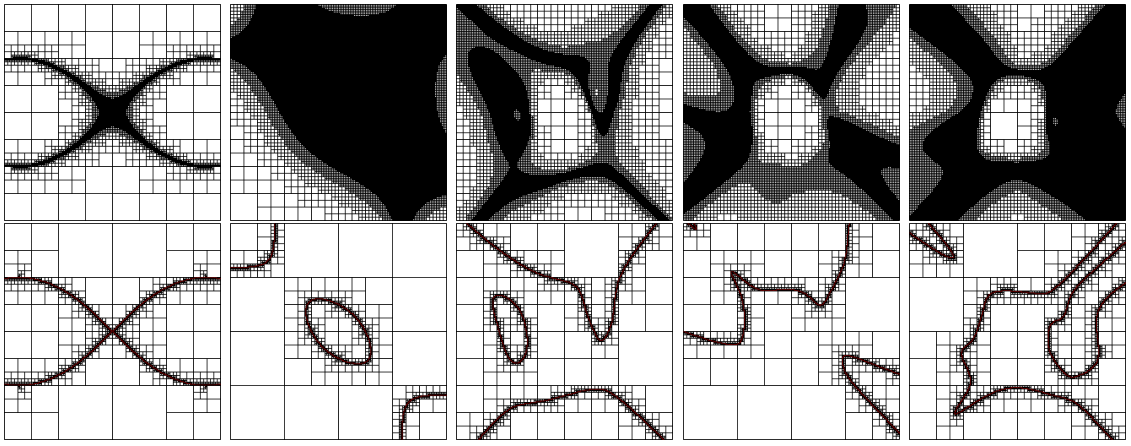

FIGURE 4.5 – En haut, les encadrements des polynômes sont calculés avec l'arithmétique d'intervalles naïve ; en bas, ils sont calculés en utilisant les bases de Bernstein. De gauche à droite, un ovale de Cassini $C_{2,2}(x,y) = 0$ dans $[-2,2] \times [-2,2]$ où $C_{a,b}(x,y) = ((x+a)^2 + y^2) \times ((x-a)^2 + y^2) - b^4$, une courbe due à Martin et al $f(x,y) = 15/4 + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0$ dans le carré $(x,y) \in [0,1]^2$, et trois courbes algébriques aléatoires de degré total 10, 14, 18.

4.4.2 Isolation de racines réelles de polynômes univariés

Les bases de Bernstein permettent à un algorithme simple et rapide d'isoler les racines réelles d'un polynôme univarié dans l'intervalle $[0,1]$. Pour cela, il faut d'abord convertir le polynôme dans la base de Bernstein, et en déduire un encadrement du polynôme. Si l'encadrement ne contient pas 0 (tous les coefficients de Bernstein ont le même signe), alors le polynôme n'a pas de racine. Sinon l'encadrement contient 0 et l'intervalle étudié est susceptible de contenir une ou des racines. Si les coefficients de Bernstein augmentent (diminuent) de façon monotone, l'intervalle contient une seule racine simple, et la méthode standard de Newton, ou la méthode de la sécante, est garantie converger vers la racine. Sinon les deux moitiés de l'intervalle en x sont étudiées récursivement, en utilisant la méthode de Casteljau pour calculer les coefficients de Bernstein des moitiés gauche et droite. Cependant, la récursion doit être arrêtée quand l'intervalle étudié en x est suffisamment petit, pour éviter de boucler indéfiniment en présence d'une racine singulière, car dans ce cas, les coefficients de Bernstein ne deviennent jamais monotones croissants ou décroissants.

Eventuellement, l'intersection de l'enveloppe convexe des points de contrôle $(i/d, b_i)$ avec l'axe des x peut être calculée : ce segment permet de contracter l'intervalle en x

étudié sans perdre ses racines éventuelles ; mais, empiriquement, cette variante n'est pas intéressante, car la subdivision et la précision des encadrements due à la base de Bernstein éliminent rapidement les intervalles en x sans racine. Nous verrons en §4.4.4 que cette optimisation est par contre intéressante dans le cas multivarié.

Pour trouver les racines de $f(x) = 0$ dans $[1, +\infty[$, on définit $g(x) = x^d f(1/x)$: si $f(x) = \sum f_i x^i$ dans la base canonique, alors $g(x) = \sum f_i x^{d-i}$, c'est-à-dire que g est un polynôme avec les mêmes coefficients que f mais dans le sens inverse. Les racines de g dans $[0, 1]$ sont à l'évidence les inverses des racines de f dans $[1, +\infty[$. De cette façon, toutes les racines positives sont calculées. Pour trouver les racines négatives de $f(x) = 0$, on calcule les racines positives du polynôme $h(x) = f(-x)$: ce sont les opposées des racines négatives de f.

Ce type d'algorithme est utilisé en infographie pour le tracé de rayons sur les surfaces algébriques implicites $f(x, y, z) = 0$ avec un degré total d, c'est-à-dire, pour calculer les points d'intersections entre un rayon (une demi-ligne) et la surface. Le rayon est paramétré avec : $x = x_0 + at$, $y = y_0 + bt$, $z_0 + z = ct$, où l'origine (x_0, y_0, z_0) du rayon est connue, ainsi que sa direction (a, b, c). Remplacer x, y, z par leurs valeurs en t donne une équation algébrique univariée en t, de degré d, qui est résolue par une variante de la méthode précédente.

4.4.3 Calcul de l'enveloppe convexe 2D

Le calcul de l'enveloppe convexe d'un ensemble de points 2D est un problème fondamental de l'infographie [81,82]. Soit (x_i, y_i) un ensemble de points dans le plan x, y. Nous expliquons seulement comment calculer la partie inférieure de l'enveloppe convexe des points (x_i, y_i) : la partie supérieure est calculée de la même manière. Premièrement, les points sont triés par x croissant. Dans notre application, c'est déjà fait puisque toutes les valeurs de x sont i/d, où d est le degré, et i un entier dans $0, ...d$. Nous supposons de plus qu'il n'y a qu'un seul point pour chaque valeur x_i, celui d'ordonnée y minimale. L'enveloppe convexe inférieure est initialisée avec les deux points de gauche $p_0 = (x_0, y_0)$; $p_1 = (x_1, y_1)$.

Ensuite, les points (x_k, y_k) sont considérés de gauche à droite (c'est-à-dire, par x_k croissant), pour $k = 2, \dots d$, et l'enveloppe convexe est mise à jour comme suit. Soit b le point le plus à droite de l'enveloppe convexe courante, et a le point juste avant dans l'enveloppe ; tant que abp_k "tourne à droite" (l'angle abp_k est concave), le point b est retiré de l'enveloppe. A la fin de cette boucle, le point p_k est inséré en fin de l'enveloppe convexe inférieure. Trois points p, q, r "tournent à droite" lorsque le déterminant de $((p_x, p_y, 1), (q_x, q_y, 1), (r_x, r_y, 1))^t$ est négatif. Quand il s'annule, les points p, q, r sont alignés ; quand il est positif, ils tournent à gauche.

Cet algorithme est en $O(d \log d)$ si d est le nombre de points, le facteur $d \log d$ est dû à la phase de tri ; comme elle est ici inutile, la méthode est linéaire en d, le nombre de points.

4.4.4 Préconditionnement et réduction

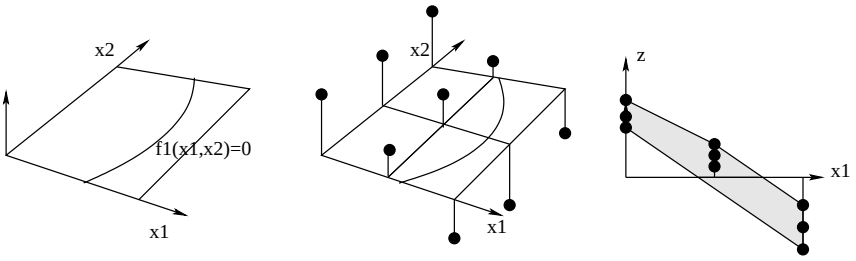

FIGURE 4.6 – L'équation $z = f_1(x_1, x_2) = 0$ est de degré 2 en x_1 et en x_2 et a une grille de points de contrôle de 3×3. La courbe $f_1(x_1, x_2) = 0$ est considérée comme l'intersection du plan $z = 0$ et de la surface $z = f_1(x_1, x_2)$. La surface est à l'intérieur de l'enveloppe convexe de ses points de contrôle $(i/2, j/2, b_{i,j}), i = 0, 1, 2, j = 0, 1, 2$. Il est facile de calculer l'enveloppe convexe 2D des projections sur le plan x_1, z des points de contrôle. L'intersection de ce polygone convexe et de l'axe x_1 est un segment qui englobe tous les x_1 des points de la courbe.

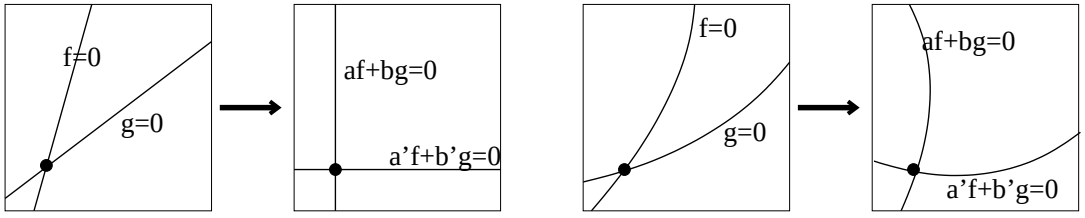

FIGURE 4.7 – Effet du préconditionnement sur un système linéaire (à gauche), sur un système non linéaire à droite.

La réduction d'un pavé, aussi appelée contraction, fournit un pavé plus petit, inclus dans le pavé étudié, et qui est garanti contenir toutes les racines. Une réduction efficace est intéressante car elle évite l'explosion combinatoire due aux subdivisions. Les subdivisions sont cependant indispensables pour séparer des racines distinctes. Les bases tensorielles de Bernstein permettent des réductions très efficaces.

Le système $f(x) = 0$ est d'abord préconditionné, pour que le jacobien du système soit la matrice identité au centre du pavé étudié. La Figure 4.7 montre un exemple 2D de préconditionnement. Le système préconditionné est $g(x) = Mf(c)$, avec $M = (f'(c))^{-1}$, $c = (u+v)/2$ le centre du pavé $[u, v]$ étudié. g et f ont les mêmes racines, et le jacobien de g en c est $g'(c) = (f'(c))^{-1} f'(c)$ est bien l'identité. En fait, une inverse approximative est suffisante. Au passage, il est possible d'éviter le calcul explicite de l'inverse, en utilisant une décomposition LU.

Le préconditionnement ne peut être appliqué facilement que si tous les f_i ont les mêmes degrés d_1 en x_1, d_2 en x_2, etc ; les points de contrôle de $g_k = \sum_i \lambda_i f_i$ sont alors les λ_i combinaisons linéaires des points de contrôle des f_i. Un prétraitement

global du système d'équations doit élever le degré des polynômes de degré inférieur.

L'élévation de degré est triviale dans la base canonique (il suffit de remplacer $x+2$ par $0x^2+x+2$, par exemple), et une simple formalité dans les bases de Bernstein [78], mais elle n'est pas anodine pour les grands systèmes (de plus de 6 ou 7 équations) : en effet, la représentation dans la base tensorielle de Bernstein a un côut exponentiel. Nous reviendrons ultérieurement sur les conséquences.

Après le préconditionnement, et pour un pavé suffisamment petit, la k ième hypersurface d'équation $g_k(x)=0$ est géométriquement proche d'un hyperplan où x_k est à peu près constant, autrement dit la coordonnée x_k des points de cette hypersurface dans le pavé étudié est comprise dans un intervalle $[u'_k, v'_k]$ de largeur plus petite que celle du pavé initial : $w_k = v_k - u_k$. C'est cet intervalle réduit $[u'_k, v'_k]$ qu'il nous faut calculer, comme suit.

Dans l'espace ($x \in \mathbb{R}^n, z \in \mathbb{R}$), chaque hypersurface $z = g_k(x)$ est incluse dans l'enveloppe convexe de ses points de contrôle : voir Figure 4.6 pour un exemple où x est dans un pavé 2D. Le calcul de cette enveloppe convexe est trivial en dimension 1, facile en dimension 2, et difficile ou impossible en grande dimension. Heureusement, l'enveloppe convexe appartient à un prisme, dont la base est la projection sur le plan (x_k, z) de ses points de contrôle. Il suffit donc de calculer ce polygone convexe avec la méthode en §4.4.3. L'intersection de ce polygone avec l'axe x_k (ou plutôt le segment $[u_k, v_k]$ sur l'axe x_k) donne l'intervalle réduit $[u'_k, v'_k]$. Si cette intersection est vide, alors le pavé ne contient pas de racine.

Mourrain et Pavone [8] utilisent une variante de cette méthode ; au lieu de calculer l'enveloppe convexe inférieure des points $(x_k, \min z_k)$, et l'enveloppe convexe supérieure des $(x_k, \max z_k)$, ils considèrent ces points comme les points de contrôle du graphe d'un polynôme dont ils calculent la plus petite et la plus grande des racines dans $[u_k, v_k]$, avec une méthode similaire à celle de §4.4.3. Ils procèdent de même pour les points $(x_k, \max z_k)$. Ils en déduisent un encadrement $[u'_u, v'_k]$ plus précis.

Quelle que soit la méthode utilisée pour réduire selon la dimension x_k, chaque dimension du pavé est réduite tour à tour, pour $k = 1, \ldots n$. Une variante possible prend en compte les réductions précédentes selon les dimensions $1, \ldots k-1$ lorsque la réduction selon l'axe x_k est effectuée.

Dans le cas où le système est linéaire, le pavé réduit est soit l'ensemble vide (la solution est hors du pavé), soit le point de la racine en supposant qu'il n'y a pas d'imprécision numérique. En pratique, un pavé réduit, de quelques ULP (*Unit in the Last Place*) de côté, et contenant le point solution, est obtenu.

Les réductions du pavé étudié sont itérées tant qu'elles réduisent significativement les pavés, par exemple tant que la longueur de chacun des n côtés est au moins divisé par 2 ou davantage.

Si après toutes ces réductions, le pavé est suffisamment petit, il est ajouté à une liste de solutions ; éventuellement, un test d'unicité sur le pavé est effectué, et le pavé est éventuellement étiqueté comme contenant une racine unique ; plusieurs solveurs permettent de "polir la racine" ou de préciser encore le pavé solution avec quelques itérations de Newton. Il se peut que le test d'unicité échoue, même pour un petit pavé : il contient une racine multiple, ou plusieurs racines simples proches, ou une quasi racine (par exemple, 0 est une quasi racine de $x^2 + 10^{-12} = 0$), et la précision finie de l'arithmétique flottante ne permet pas de décider.

Si le pavé reste gros et ne peut plus être significativement réduit, il contient vraisemblablement plusieurs racines. Le pavé doit être coupé en deux, par exemple selon son côté le plus long, ou bien selon le côté qui a été le moins réduit (en valeur relative) lors de la dernière réduction effectuée. Les deux pavés sont empilés. Il est aussi possible de couper en deux selon plusieurs dimensions, 2, 3, ou 4, ce qui donne 2^2, 2^3, 2^4 sous pavés.

4.5 Quelques tests utiles

4.5.1 Preuve qu'un pavé ne contient pas de racine

Parfois, la méthode de réduction ne peut pas détecter rapidement qu'un pavé ne contient pas de racine. Par exemple, en 2D, pour deux courbes "parallèles" (*offset*) et proches, il faut subdiviser le long des deux courbes jusqu'à les séparer, ce qui est côuteux. Le test suivant détecte très vite ce genre de situation.

Soit $f_1(x) = f_2(x) = ... = f_n(x) = 0$ un système d'équations. Nous supposons que tous les f_i ont les mêmes degrés pour toutes les variables. S'il existe des nombres $\lambda_i \in \mathbb{R}^n$ tels que $g(x) = \sum_i \lambda_i f_i(x)$ a tous ses coefficients de Bernstein strictement positifs, par exemple supérieurs ou égaux à 1, alors $g(x)$ est toujours positif dans le pavé étudié ; or g s'annule pour toutes les racines communes des f_i. Ceci prouve donc que les f_i n'ont aucune racine commune dans le pavé. Les coefficients de Bernstein de g sont les λ_i combinaisons linéaires des coefficients de Bernstein des f_i, si bien que les λ_i existent si le problème correspondant de programmation linéaire a une solution.

Cette idée simple s'étend aux systèmes d'équations et aux inéquations. Une illustration en est donnée Figure 4.8. Supposons que le problème est de trouver x dans un pavé de R^n tel que $f_1(x) = f_2(x) = \ldots = f_n(x) = 0$ et que $g_1(x) \leq 0, \ldots g_m(x) \leq 0$. S'il existe des $\lambda_i \in R^n$ et des $\mu_j \geq 0$ tels que $h = \sum_i \lambda_i f_i + \sum_j \mu_j g_j \geq 1$ (le dernier 1 peut être remplacé par n'importe quelle constante positive) pour tous les points x du pavé, alors le système n'a pas de solution. En effet, considérons x^* une racine commune des f_i. Alors comme $h(x^*) \geq 1$, il vient $h(x^*) = \sum_i \lambda_i f_i(x^*) + \sum_j \mu_j g_j(x^*) = \sum_j \mu_j g_j(x^*) \geq 1 > 0$; mais ceci contredit les contraintes $g_j(x) \leq 0$. Trouver si les λ_i et les μ_j existent est un problème de programmation linéaire.

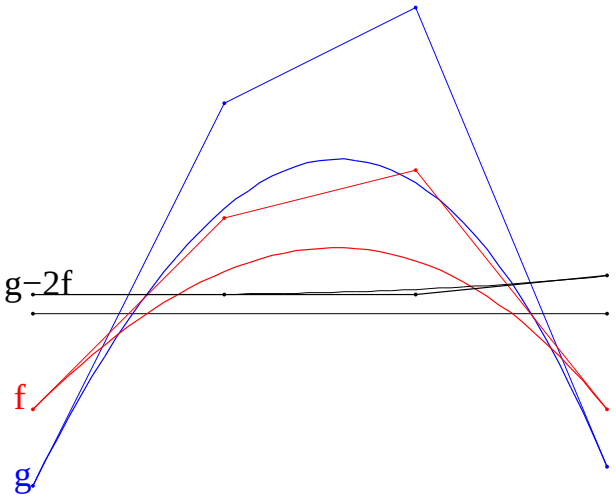

FIGURE 4.8 – Le problème est de trouver x tel que $f(x) = 0$ et $g(x) \leq 0$. Dans cet exemple, les coefficients de Bernstein de $g - 2f$ sont tous strictement positifs. Donc $g - 2f$ est strictement positif sur l'intervalle considéré. $g(x^*) - 2f(x^*) > 0$ et $f(x^*) = 0 \Rightarrow g(x^*) > 0$: donc il est prouvé que le système $f(x) = 0$ et $g(x) \leq 0$ n'a pas de solution dans l'intervalle.

4.5.2 Preuve d'existence d'une racine

Le théorème de Miranda, aussi appelé théorème de Poincaré-Miranda fournit un test d'existence d'une racine dans un pavé. Il généralise le théorème de Rolle, dit des valeurs intermédiaires, au cas multidimensionnel. Soit f_i, $i = 1, \ldots n$ fonctions continues de R^n vers R ; soit un pavé $[u, v]$ de R^n, $u = (u_i), v = (v_i)$. Si pour tout k dans $1, \ldots n$, la fonction f_k est de signe constant sur la face $x_k = u_k$, et de signe constant mais opposé sur la face opposée du pavé $x_k = v_k$, alors il existe au moins une racine commune aux f_i dans le pavé. Une fonction f_k est de signe constant sur la face $x_k = u_k$ si l'encadrement de f_k sur le sous pavé correspondant (l'intervalle pour x_k est $[u_k, u_k]$ au lieu de $[u_k, v_k]$) ne contient pas 0. On remarquera que, pour un pavé suffisamment petit contenant une racine r commune aux f_i, le préconditionnement produit des hypersurfaces d'équation $g_k(x) = 0$ géométriquement très proches de l'hyperplan $x_k - r_k = 0$, et les conditions du théorème sont donc probablement vérifiées pour le système préconditionné.

4.5.3 Preuve d'unicité d'une racine dans un pavé

Elbert et Kim [9] ont proposé un test pour décider si un pavé contient une seule racine simple (une racine est simple si le jacobien est de rang n). Ce test est bien adapté aux bases géométriques, comme les bases tensorielles des splines ou de Bernstein. Soit T_i le cône des vecteurs tangents aux hypersurfaces $z = f_i(x)$ dans le pavé étudié. Elbert et Kim encadrent ces cônes avec un vecteur axial (la moyenne des vecteurs générateurs) et un intervalle angulaire. Quand le vecteur nul est le seul vecteur commun à tous les T_i, alors le pavé étudié contient au plus une racine commune aux f_i. Après le préconditionnement, quand le pavé considéré contient une seule

racine simple r, cette condition est probablement vérifiée, car le préconditionnement tend à créer des hypersurfaces $g_k(x) = 0$ géométriquement proches des hyperplans d'équation $x_k - r_k = 0$ (Figure 4.7).

Elbert et Kim stoppent la subdivision dès que l'existence et l'unicité sont prouvées, et recourent alors à quelques itérations de Newton pour préciser la racine, en partant du centre du pavé. Cette optimisation peut accélérer quelque peu le solveur, mais la complexité théorique du solveur reste inchangée.

Un autre théorème donnant un test d'unicité est le théorème de Newton-Kantorovitch. Il est utilisé dans le solveur ALIAS [77]. Comme ce théorème utilise des encadrements des dérivées secondes des équations, il est très commode pour les systèmes quadratiques où toutes les dérivées secondes sont constantes.

4.6 Les limitations de ce premier type de solveur

Ce type de solveur est très satisfaisant pour les systèmes assez petits qui sont rencontrés en infographie : le calcul des points d'intersection d'une droite avec une surface algébrique implicite d'équation $f(x, y, z) = 0$, ou avec une surface paramétrique polynomiale d'équations : $(x = X(u, v), y = Y(u, v), Z = Z(u, v))$ ou une surface paramétrique rationnelle d'équations : $H(u, v) \times x = X(u, v)$, $H(u, v) \times y = Y(u, v)$, $H(u, v) \times z = Z(u, v)$, où $(u, v) \in [0, 1]^2$, et X, Y, Z, H sont des polynômes, souvent déjà donnés dans la base de Bernstein, ou des fonctions polynomiales par morceaux souvent données dans une base de splines, comme dans [9]. A raison de 24 ou 25 images par secondes, de un million ou plus de pixels pour chaque image, et de quelques dizaines de telles surfaces dans la scène, le calcul de film d'animation par la méthode du lancer de rayons permet de tester de façon intensive ce type de solveurs. Une autre application, moins intensive, est le calcul des points d'intersection entre trois des surfaces précédentes.

Ce type de solveur se heurte à deux difficultés. La première difficulté est due à l'imprécision numérique, et abordée en §4.6.1 et facilement surmontée. La seconde est plus délicate, et due au côut de la représentation dans la base tensorielle de Bernstein, et est abordée en §4.6.2.

4.6.1 Imprécision numérique

Les difficultés dues à l'imprécision numérique ne sont pas propres à ce solveur.

Une première solution serait d'ignorer le problème, et donc effectuer les calculs en arithmétique flottante. Cette approche ne peut être justifiée que pour des applications non critiques, comme la synthèse d'images, où l'imprécision aura pour principal effet, imperceptible, de déplacer d'un pixel les contours apparents des objets. Mentionnons cependant que quelques précautions non complètement triviales sont nécessaires pour ne pas manquer la racine dans le cas de systèmes linéaires.

Une seconde solution représente les coefficients de Bernstein par des intervalles ; c'est ce que font Hu et Patrikalakis par exemple. La largeur de ces intervalles est initialement de un ou deux ULP, et croît, relativement lentement, au fur et à mesure des subdivisions de Casteljau. Il vient un moment où subdiviser devient inutile : ainsi, en supposant que les bornes des intervalles sont des entiers (les nombres flottants formant eux aussi un ensemble discret), la moitié gauche et la moitié droite de l'intervalle $[0, 1]$ sont toutes deux $[0, 1]$; plus généralement, la largeur des deux moitiés gauche et droite d'un intervalle de largeur 1 est 1. Cette seconde approche peut poser quelques difficultés techniques, lorsqu'est utilisée la programmation linéaire, comme en §4.5.1. La mise à l'échelle présentée dans le paragraphe §5.4.2 peut traiter ce problème.

4.6.2 Coût exponentiel de la représentation

Ces solveurs ont une limitation plus fondamentale, qui devient critique quand le nombre n d'inconnues et d'équations augmente, et devient supérieur à 6 ou 7 en pratique. Ces solveurs calculent tous les coefficients de Bernstein des polynômes, pour trouver le plus petit coefficient, et le plus grand. Or, si les polynômes sont creux dans la base tensorielle canonique, ils cessent de l'être dans la base tensorielle de Bernstein. Par exemple, le monôme 1 de la base tensorielle canonique est dense dans la base tensorielle de Bernstein ; il s'écrit :

$$1 = (\ B_0^{(d_1)}(x_1) + \ldots\ B_{d_1}^{(d_1)}(x_1)) \times \ldots \times (\ B_0^{(d_n)}(x_n) + \ldots\ B_{d_n}^{(d_n)}(x_n))$$

dans la base tensorielle de Bernstein. Pour un système linéaire en n inconnues, il nécessite 2^n coefficients, tous égaux à 1. Pour un système quadratique (chaque variable apparaît avec un degré au plus 2), il nécessite 3^n coefficients ; si en chaque variable apparaît avec un degré au plus d, il nécessite $(d+1)^n$ coefficients, tous égaux à 1. Ainsi le plus simple des monômes : 1, a une représentation dans la base tensorielle de Bernstein qui a une taille exponentielle. En conséquence, le calcul de l'encadrement des valeurs d'un polynôme dans un pavé, et la méthode de réduction, nécessitent un temps exponentiel, si bien que ces solveurs sont inutilisables pour les grands systèmes.

Comment lever cette limitation ? Il se trouve que, parmi une quantité exponentielle de coefficients de Bernstein, seuls le plus petit et le plus grand sont utiles : ils donnent les bornes des encadrements. Ne pourrait-on calculer uniquement ces deux coefficients ? Le nouveau solveur présenté utilise cette remarque de bon sens, en recourant à la programmation linéaire (ce que font déjà certains des solveurs du premier type, avec le test proposé en §4.5.1).

4.7 Conclusion

Dans cette première partie nous avons revu la définition et les propriétés essentielles des bases tensorielles de Bernstein, ainsi que les conversions entre ces bases et la base canonique, avant de présenter les applications de ces bases pour l'isolation des racines des polynômes univariés et multivariés, et de discuter les difficultés

des solveurs utilisant ces bases. Ces difficultés sont dues à l'imprécision numérique d'une part et au coût de la représentation de polynômes dans la base tensorielle de Bernstein d'autre part.

Chapitre 5

Solveur et polytope de Bernstein

Cette partie présente un nouveau type de solveur, qui utilise de la programmation linéaire sur un polytope. Il est défini en §5.1. Pour simplifier, notre travail considère des systèmes quadratiques : tous les monômes des polynômes du système sont de degré total au plus 2. Ceci est sans perte de généralité. D'une part, beaucoup de systèmes en modélisation géométrique par contraintes se réduisent à la résolution de systèmes quadratiques ; d'autre part, plusieurs méthodes permettent de réduire tout système algébrique à un système quadratique ; c'est du reste ce qui est fait dans l'article [83] sur des monômes de degré 4. Enfin, il est aussi possible de généraliser le polytope de Bernstein à des degrés supérieurs ; si d est le degré total maximum des monômes, le polytope a $O(n^d)$ hyperplans et hyperfaces ; le nombre de sommets du polytope, par contre, est toujours exponentiel, plus grand que 2^n.

C. Fünfzig *et al.* ont réalisé un prototype de ce solveur [83]. Il résout des systèmes de taille arbitraire, par exemple avec 60 inconnues et équations, générées en calculant le pavage de cercles représentant un graphe planaire complètement triangulé. Il résout aussi des systèmes non quadratiques.

Cette partie est structurée ainsi : §5.1 définit le polytope de Bernstein. Ensuite, §5.2 montre comment l'optimisation d'une fonction objectif linéaire sur le polytope de Bernstein, en recourant à la programmation linéaire, permet de calculer des encadrements des valeurs d'un polynôme dans un pavé (§5.2.1), et de réduire un pavé tout en préservant les racines qu'il contient (§5.2.2). Les principales caractéristiques du nouveau solveur sont ensuite présentées en §5.3. Ce solveur ne fait quasiment plus référence aux bases tensorielles de Bernstein. §5.4 discute quelques détails d'implantation à savoir la mise à l'échelle en §5.4.1 et surtout comment prévenir les erreurs dues à l'imprécision numérique de l'arithmétique flottante en §5.4.2.

5.1 Définition du polytope de Bernstein

L'idée essentielle est d'encadrer l'ensemble semi-algébrique (*patch*) :

$$S = (x_1, x_2, \ldots x_n, x_1^2, x_1x_2, \ldots x_1x_n, x_2^2, \ldots x_2x_n, \ldots x_n^2), x \in [0,1]^n$$

par un polytope (un polyèdre convexe) de $\mathbb{R}^N$, $N = n(n+3)/2$ défini par ses hyperplans. S et ce polytope vivent dans un espace à N dimensions, une dimension

par monôme (à l'exception du monôme 1). D. Michelucci a appelé ce polytope le polytope de Bernstein, car ses hyperplans sont les inéquations : $B_i^{(2)}(x_k) \geq 0$ avec $i = 0, 1, 2$, ou des inéquations : $B_i^{(1)}(x_k)\ B_j^{(1)}(x_l) \geq 0$ avec $i, j \in [0, 1]^2$.

Ultérieurement, chaque monôme sauf 1 sera associé à une variable d'un problème de programmation linéaire, par exemple les monômes $x_i, x_i^2, x_i x_j$ seront associés aux variables X_i, X_{ii}, X_{ij}. Il faudra optimiser une fonction objectif linéaire en les variables X_i, X_{ii}, X_{ij}. Détaillons d'abord les hyperplans du polytope. Soit x une variable parmi $x_1, \ldots x_n$. L'arc de courbe (x, x^2) avec $0 \leq x \leq 1$ est encadré par un triangle, Figure 5.1. Soit y la variable représentant le monôme x^2, les inéquations définissant ce triangle sont :

$$B_0^{(2)}(x) = (1-x)^2 = x^2 - 2x + 1 \geq 0 \Rightarrow y - 2x + 1 \geq 0$$
$$B_1^{(2)}(x) = 2x(1-x) = -2x^2 + 2x \geq 0 \Rightarrow -2y + 2x \geq 0$$
$$B_2^{(2)}(x) = x^2 \geq 0 \Rightarrow y \geq 0$$

Il y a au total $3n$ de ces hyperplans. Soit x_i et x_j deux variables distinctes parmi $x_1, \ldots x_n$. Il y a $n\,(n-1)\,/2$ telles paires. Par commodité, on les renomme x et y, et le monôme xy est représenté par une variable z.

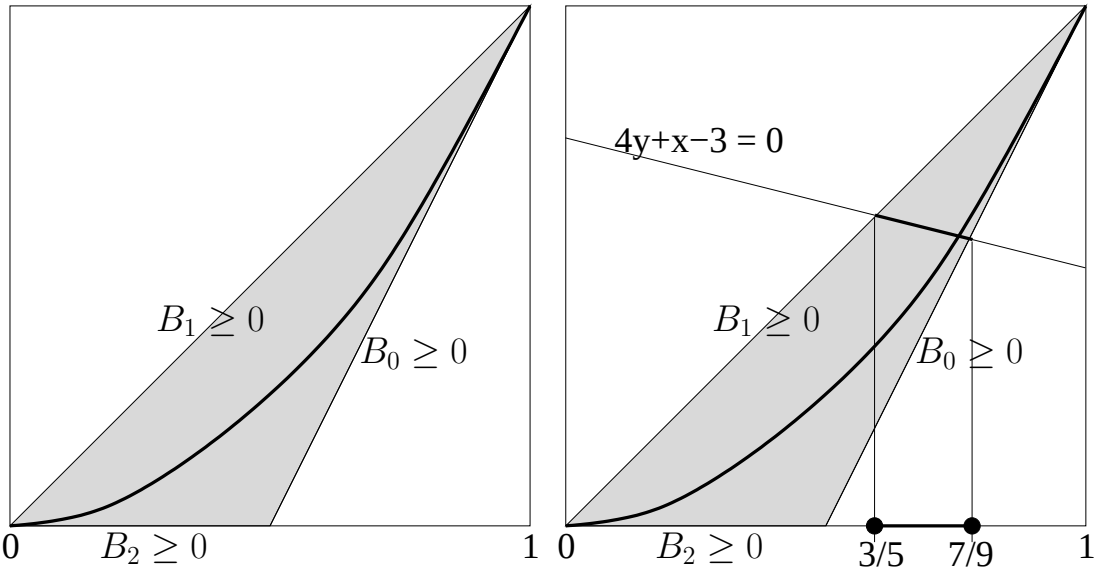

FIGURE 5.1 – (a) : le polytope de Bernstein encadrant la courbe $(x, y = x^2)$, pour $(x, y) \in [0, 1]^2$. C'est l'intersection des demi-plans d'équations : $B_0(x) = (1-x)^2 = y - 2x + 1 \geq 0$, $B_1(x) = 2x(1-x) = 2x - 2y \geq 0$, $B_2(x) = x^2 = y \geq 0$. A droite : résoudre $4x^2 + x - 3 = 0$, avec $x \in [0, 1]$, équivaut à calculer l'intersection de la droite $4y + x - 3 = 0$ avec la courbe (x, x^2). La programmation linéaire donne l'intersection entre la droite et le polytope de Bernstein : l'intervalle en x, initialement $[0, 1]$, est réduit à $[3/5, 7/9]$.

.

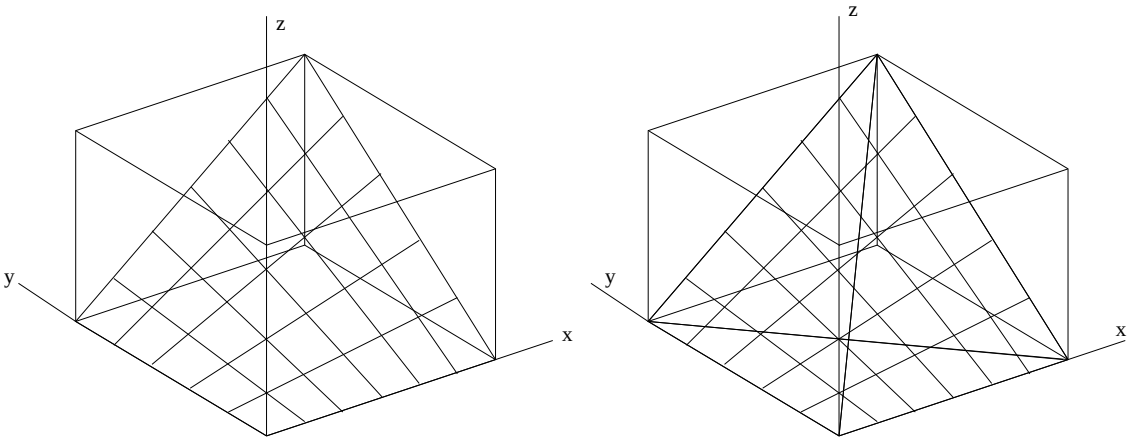

FIGURE 5.2 – Le polytope de Bernstein encadrant le carreau de surface, un PH : $(x, y, z = xy)$. Les inéquations des plans des 4 faces sont : $B_i(x)B_j(y) \geq 0$, où $i = 0, 1$ et $B_0(t) = 1 - t$ et $B_1(t) = t$.

Dans l'espace 3D (x, y, z), la surface contenant les points $(x, y, z = xy), x \in [0, 1], y \in [0, 1]$ est un morceau de paraboloïde hyperbolique dont l'enveloppe convexe est un tétraèdre, Figure 5.2. Les inéquations des plans de ce tétraèdre sont :

$$\begin{aligned}
&B_0^{(1)}(x)\ B_0^{(1)}(y) = (1-x)(1-y) = 1 - x - y + xy \geq 0 \\
&\quad \Rightarrow 1 - x - y + z \geq 0 \\
&B_0^{(1)}(x)\ B_1^{(1)}(y) = (1-x)y = y - xy \geq 0 \Rightarrow y - z \geq 0 \\
&B_1^{(1)}(x)\ B_0^{(1)}(y) = x(1-y) = x - xy \geq 0 \Rightarrow x - z \geq 0 \\
&B_1^{(1)}(x)\ B_1^{(1)}(y) = xy \geq 0 \Rightarrow z \geq 0
\end{aligned}$$

Il y a au total $2n(n-1)$ tels hyperplans.

Au total, le polytope de Bernstein a $3n + 2n(n-1) = n(2n+1)$ hyperplans dans un espace à $N = n(n+3)/2$. En fonction du nombre n d'inconnues du système d'équations, il a un nombre polynomial $O(n^2)$ d'hyperplans et donc de faces. Par contre il a une quantité exponentielle de sommets dans le cas multivarié $(n > 1)$. En effet, sur l'espace $(x_1, x_2, \ldots x_n)$, le polytope de Bernstein se projette selon l'hypercube $[0, 1]^n$, qui a 2^n sommets ; le polytope de Bernstein a donc au moins autant de sommets (la projection d'un polyèdre convexe ne peut pas avoir plus de sommets que le polyèdre initial).

En pratique, d'autres hyperplans sont utilisés pour réduire encore le polytope. Pour toute paire d'inconnues x, y différentes, est ajoutée l'inéquation : $(x - y)^2 \geq 0 \Rightarrow x^2 + y^2 - 2(xy) \geq 0$: en remplaçant les monômes x^2, y^2 et xy par les variables correspondantes, cela donne une inéquation linéaire dans $\mathbb{R}^N$, i.e., un hyperplan qui tronque le polytope de Bernstein. De plus, pour chaque inconnue x parmi $x_1, \ldots x_n$ l'inéquation : $(x - 1/2)^2 \geq 0 \Rightarrow x^2 - x + 1/4 \geq 0$ est ajoutée (Figure 5.4).

Cette liste récapitule les inéquations des hyperplans (X_i est la variable du monôme x_i, X_{ii} celle de x_i^2, X_{ij} celle de $x_i x_j$) :

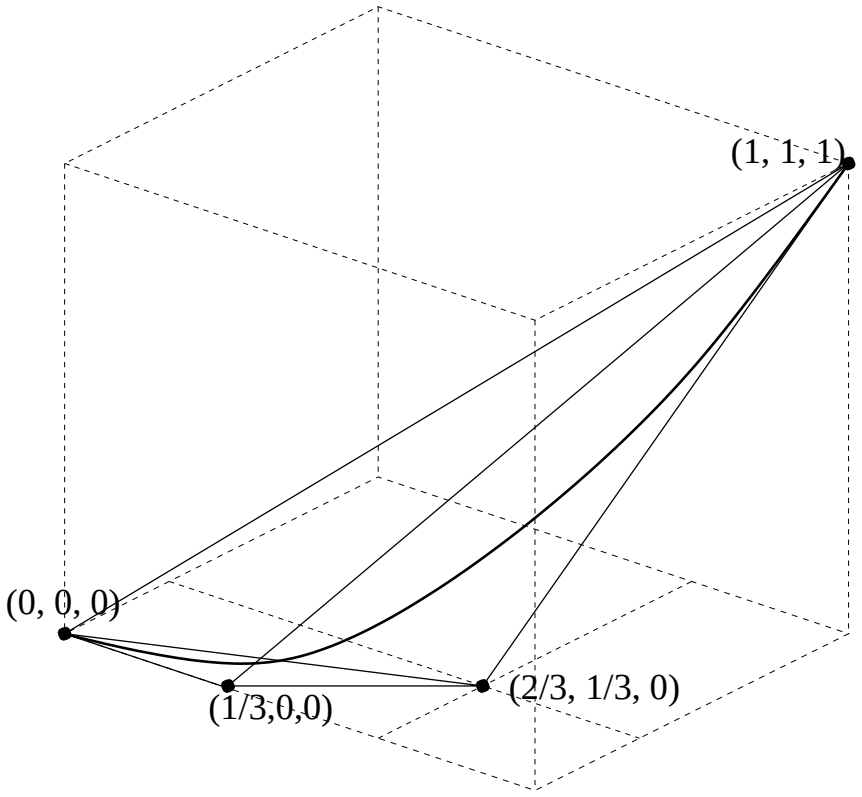

FIGURE 5.3 – Le polytope de Bernstein, un tétraèdre, encadrant la courbe $(x, y = x^2, z = x^3)$ où $x \in [0,1]$. Ses sommets sont : $v_0 = (0,0,0)$, $v_1 = (1/3,0,0)$, $v_2 = (2/3,1/3,0)$ et $v_3 = (1,1,1)$. v_0 est sur le plan $B_1 = B_2 = B_3 = 0$, v_1 sur $B_0 = B_2 = B_3 = 0$, etc. Le tétraèdre est l'intersection des demi-espaces : $B_0(x) = (1-x)^3 = 1-3x+3x^2-x^3 \geq 0 \Rightarrow 1-3x+3y-z \geq 0$, $B_1(x) = 3x(1-x)^2 = 3x-6x^2+3x^3 \geq 0 \Rightarrow 3x-6y+3z \geq 0$, $B_2(x) = 3x^2(1-x) = 3x^2-3x^3 \geq 0 \Rightarrow 3y-3z \geq 0$, $B_3(x) = x^3 \geq 0 \Rightarrow 3z \geq 0$.

$0 \leq X_{ii} - 2X_i + 1$	$1 \leq i \leq n$	triangle
$0 \leq -2X_{ii} + 2X_i$	$1 \leq i \leq n$	triangle
$0 \leq X_{ii}$	$1 \leq i \leq n$	triangle
$0 \leq 1 - X_i - X_j + X_{ij}$	$1 \leq i < j \leq n$	tétraèdre
$0 \leq X_i - X_{ij}$	$1 \leq i < j \leq n$	tétraèdre
$0 \leq X_j - X_{ij}$	$1 \leq i < j \leq n$	tétraèdre
$0 \leq X_{ij}$	$1 \leq i < j \leq n$	tétraèdre
$0 \leq X_{ii} + X_{jj} - 2X_{ij}$	$1 \leq i < j \leq n$	complément
$0 \leq X_{ii} - X_i + 1/4$	$1 \leq i \leq n$	complément

Pour $n = 2$, le volume du polytope est approximativement 0.007, alors que le volume de l'hypercube en $N = 5$ dimensions est 1. Ce ratio des volumes décroît exponentiellement quand n augmente. Ce faible volume du polytope explique l'efficacité du solveur, quand on le compare à ceux qui utilisent l'arithmétique d'intervalles naïve : cette dernière encadre S par l'hypercube, bien plus gros que le polytope de Bernstein. Le polytope de Bernstein se généralise à des degrés $d > 2$ plus élevés ; pour

des monômes de degré maximal d, le polytope a $O(n^d)$ hyperplans, et un nombre exponentiel de sommets dans le cas multivarié, au moins 2^n, pour les mêmes raisons que dans le cas quadratique. La Figure 5.3 montre le polytope de Bernstein pour le cas univarié de degré 3 : il s'agit d'un tétraèdre.

5.2 Le recours à la programmation linéaire

5.2.1 Calcul d'encadrements

Calculer un encadrement de l'image de l'hypercube $[0,1]^n$ par un polynôme donné se réduit à un problème de programmation linéaire. La méthode est illustrée sur un exemple simple. Pour calculer un encadrement de $p(x) = 4x^2 + x - 3$, pour $x \in [0,1]$, la fonction linéaire objectif : $4y + x - 3$ est minimisée, puis maximisée, sur le polytope de Bernstein (le triangle en Figure 5.1) qui contient la courbe $(x, y = x^2), x \in [0,1]$. C'est un problème de programmation linéaire, après avoir remplacé le monôme x^2 par y. De gauche à droite, voici les tableaux de la méthode du simplexe pour le problème, pour le minimum, et pour le maximum. Les variables d'écart sont appelées b_0, b_1, b_2, en lien avec les polynômes de Bernstein $B_0,\ B_1,\ B_2$; par exemple B_0 est remplacé par la variable b_0.

Les tableaux des deux problèmes sont :

$$\begin{aligned}
&\min, \max : p = 4y + x - 3 \\
&0 \leq b_0 = y - 2x + 1 \\
&0 \leq b_1 = -2y + 2x \\
&0 \leq b_2 = y
\end{aligned}$$

Les tableaux solutions sont :

$$\begin{array}{ll}
\min p = -3 + x + 4y & \max p = 2 - 5b_0 - 9/2b_1 \\
b_0 = 1 - 2x + y & x = 1 - b_0 - b_1/2 \\
b_1 = 2x - 2y & y = 1 - b_0 - b_1 \\
b_2 = y & b_2 = 1 - b_0 - b_1
\end{array}$$

La méthode du simplexe, due à George Dantzig en 1947 [84], effectue des pivots de Gauss sur les lignes du tableau initial, pour atteindre les tableaux finaux du minimum et du maximum. Seul le tableau du maximum est commenté. Dans $\max p = 2 - 5b_0 - 9/2b_1$, la valeur de p ne peut excéder 2 : en considérant que les variables de droite b_0 et b_1 sont nulles, les accroître ne peut que diminuer p à cause des coefficients négatifs $-5b_0 - 9/2b_1$ dans la fonction objectif ; et par ailleurs, toutes les variables, en particulier b_0 et b_1 doivent être positives. Les mêmes considérations s'appliquent pour le minimum $p = -3 + x + y$, qui est atteint pour $x = y = 0$: x et y peuvent seulement être augmentées, mais cela fait augmenter p à cause des coefficients positifs de x et y dans l'expression $p = -3 + x + y$. Quand le minimum (le maximum) se produit en un sommet extrêmal (c'est à dire "au dessus" d'un sommet de l'hypercube $[0,1]^n$), il est exact. Dans l'exemple, le minimum se produit en $x = 0$,

et le maximum en $x = 1$, qui sont les 2 sommets de l'"hypercube" $[0,1]^1$; ils sont donc exacts.

On remarque qu'au sommet $v_0 = (0,0)$ où $b_1 = b_2 = 0$, la valeur de la fonction objectif p est $p_0 = p(0) = -3$; au sommet $v_1 = (1/2, 0)$ où $b_0 = b_2 = 0$, p vaut $p_1 = p(1/2) = -3/2$; au sommet $v_2 = (1,1)$ où $b_0 = b_1 = 0$, p vaut $p_2 = p(1) = 2$. Ces valeurs p_0, p_1, p_2 sont les coefficients dans la base de Bernstein du polynôme $p(x) = p_0\ B_0(x) + p_1\ B_1(x) + p_2\ B_2(x)$.

Cette propriété s'étend à tous les polynômes univariés, en admettant bien sûr que seules soient utilisées les inégalités de Bernstein $B_i^{(d)}(x) \geq 0$; ceci est dû au fait que le polytope de Bernstein est un simplexe dans le cas univarié.

Un précurseur de cette méthode de calcul d'encadrement est Olivier Beaumont [85] ; dans sa thèse, il utilise les polynômes de Chebychev et la programmation linéaire pour encadrer des polynômes multivariés.

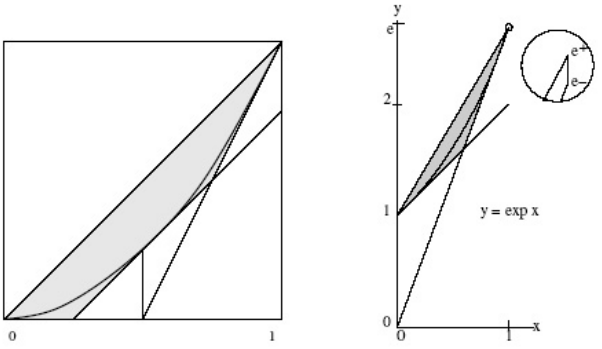

FIGURE 5.4 – Gauche : il est possible de réduire encore le polytope de Bernstein, *e.g.,* avec l'inégalité $x - y \leq 1/4$. Droite : un polygone convexe enclôt l'arc de courbe $(x, y = \exp x), 0 \leq x \leq 1$. $e = \exp 1$ est inclus dans l'intervalle $[e^-, e^+]$, où e^- et e^+ sont deux nombres flottants successifs.

5.2.2 Réduction préservant les racines

Ce paragraphe montre comment le solveur réduit les intervalles des inconnues x_i tout en préservant les solutions contenues dans ces intervalles.

Résoudre l'équation $4x^2 + x - 3 = 0$ avec $x \in [0,1]$ équivaut à trouver les points d'intersection entre la droite d'équation $4y + x - 3 = 0$ et l'arc de courbe $y = x^2$, $x \in [0,1]$. Cet arc de courbe est inscrit dans son polytope de Bernstein, le triangle de la Figure 5.1. Calculer l'intersection entre la droite et ce triangle, i.e., trouver la valeur minimale et maximale de x de cette intersection, sont des problèmes de programmation linéaire. Les problèmes sont presque les mêmes que

dans le paragraphe précédent, seuls changent les fonctions objectifs ; ce coup ci, on souhaite minimiser et maximiser x. Les tableaux du simplexe sont :

$\min, \max : x$	$\min x = 3/5 + 2/5b_1$	$\max x = 7/9 - 4/9b_0$
$0 \leq b_0 = y - 2x + 1$	$y = 3/5 - 1/10b_1$	$y = 5/9 + 1/9b_0$
$0 \leq b_1 = -2y + 2x$	$b_0 = 2/5 - 9/10b_1$	$b_1 = 4/9 - 10/9b_0$
$0 \leq b_2 = y$	$b_2 = 3/5 - 1/10b_1$	$b_2 = 5/9 + 1/9b_0$

Donc l'intervalle $[0, 1]$ pour x est réduit à $[3/5, 7/9]$, et aucune racine n'est perdue. Pour réduire davantage l'intervalle, il faut utiliser une mise à l'échelle détaillée en §5.4.1 qui applique l'intervalle $x \in [3/5, 7/9]$ sur $x' \in [0, 1] : x = 3/5 + (7/9 - 3/5)x' = b + ax'$, et l'équation en fonction de x' est : $4a^2x'^2 + (8ab + b)x' + (b - 3) = 0$. La convergence autour d'une racine régulière est super quadratique, comme pour la méthode de Newton-Raphson. Informellement, l'argument est le suivant : si le système est linéaire, une seule réduction suffit, et le pavé réduit est égal au point solution ; or plus le pavé contenant une seule solution régulière est réduit, et plus le système est proche d'un système linéaire : géométriquement, les hypersurfaces deviennent de plus en plus proches de leurs hyperplans tangents. Une preuve mathématique de la vitesse de convergence est donnée par Mourrain et Pavone [8] ; cette preuve s'applique aux deux solveurs . Une conséquence importante de la vitesse de convergence est qu'une pile de petite taille est suffisante.

Remarque : si la droite de l'exemple ne coupe pas le triangle de Bernstein, ou plus généralement si le problème de programmation linéaire n'est pas réalisable, ceci prouve que le pavé considéré ne contient pas de racine.

5.3 Caractéristiques du nouveau solveur

Le polytope de Bernstein permet de proposer un nouveau solveur, qui ne se réfère plus à la méthode de Newton par intervalles, ni même aux bases de Bernstein.

Dans ce nouveau solveur, l'ensemble $S = (x_k, x_i x_j)$, $i \in [1, n], j \in [1, n-1], k \geq j$, $x_i \in [0, 1]^n$ dans $\mathbb{R}^N$ est encadré comme décrit précédemment par un polytope. De plus chaque équation du système (*a priori* non linéaire) en x_i, $i \in [1, n]$ donne une équation linéaire qui décrit un hyperplan de $\mathbb{R}^N$ quand chaque monôme est remplacé par la variable correspondante. De même les inéquations algébriques du système, s'il y en a, sont converties en inéquations linéaires. Cette méthode prend donc en compte très facilement les inéquations algébriques.

Comme les solveurs classiques par intervalles, le nouveau solveur gère une pile de pavés à étudier. La pile est initialisée avec le pavé où on souhaite trouver les solutions du système. Tant que la pile est non vide, le pavé en sommet de pile est dépilé. Ce pavé est réduit, en réduisant l'intervalle pour chaque inconnue x_i ; au total, $2n$ optimisations linéaires sont donc effectuées : minimiser et maximiser x_i pour i de 1 à n. Si le pavé réduit est vide, alors il est garanti ne pas contenir de racine. S'il est non vide et suffisamment petit, alors il est inséré dans une liste des solutions (il est possible de garantir qu'un pavé contient une seule racine, mais cette procédure n'est

pas spécifique à ce solveur [9]). S'il est non vide et significativement réduit, alors une autre réduction est effectuée. Sinon, il contient probablement plusieurs racines, et le couper en 2 est le seul moyen de séparer les racines ; le pavé est donc coupé en 2, par exemple selon l'inconnue x_i dont l'intervalle est le plus grand, ou bien selon l'inconnue x_i dont l'intervalle a été le moins réduit, et les deux pavés moitiés sont empilés. Plusieurs définitions pour "significativement réduit" sont possibles, et plusieurs tests de terminaison peuvent être imaginés ; cette question n'est pas détaillée ici.

Un avantage de ce solveur est que le préconditionnement du système, et l'inversion du jacobien, deviennent inutiles : la programmation linéaire fait l'essentiel du travail. Ce solveur est donc très simple, conceptuellement, et facile à programmer.

La Figure 5.1 à droite montre l'application de cette méthode sur l'équation $4x^2 + x - 3 = 0$, avec $x \in [0, 1]$. Soit y la variable représentant le monôme x^2. Alors l'intersection du polytope de Bernstein et de la droite d'équation $4x^2 + x - 3 = 0$ permet de réduire l'intervalle pour x à $[3/5, 7/9]$ sans perdre de racines. Ensuite cette intervalle est appliqué sur $[0, 1]$ par la mise à l'échelle : $x = 3/5 + (7/9 - 3/5)x'$, $x' \in [0, 1]$ (§5.4.1).

La Figure 5.5 montre des exemples 2D, donc pour des systèmes avec 2 inconnues et 2 équations. Tous les pavés sont dessinés, et représentés par des rectangles, aux côtés horizontaux ou verticaux. Le cadre de chaque sous figure est le pavé initial. Les points solutions sont représentés par un petit disque noir ; le pavé visible le plus petit qui contient ce disque noir est réduit en une opération de réduction à un pavé plus petit que le disque, et donc non visible sur la figure. Ceci illustre la convergence super-quadratique du solveur, pour les solutions régulières. Dans le cas d'une solution singulière (le point d'intersection entre deux coniques tangentes entre elles), la convergence n'est plus que linéaire, mais les réductions demeurent suffisamment significatives pour que le pavé ne soit pas coupé en deux. Les pavés sans solution sont détectés très vite.

Ce solveur a été utilisé dans [83] pour résoudre des contraintes géométriques en 3D : la plateforme de Gough-Stewart (ou octaèdre), le calcul des droites tangentes à quatre sphères données, le calcul de pavages de cercles, etc. Ce dernier problème génère des systèmes de taille arbitrairement grande, qui sont insolubles avec le type classique des solveurs de Bernstein .

5.4 Détails de réalisation

5.4.1 Mise à l'échelle

Un pavé réduit n'est plus $[0, 1]^n$. Une mise à l'échelle est nécessaire pour appliquer le pavé réduit $[u, v]$, où $u_i \leq v_i$, sur l'hypercube unité $[0, 1]^n$: soit $x_i = u_i + (v_i - u_i)x'_i$, $w_i = v_i - u_i$, $x' \in [0, 1]^n$. Alors $x_i^2 = w_i^2 x_i'^2 + 2u_i w_i x'_i + u_i^2$, $x_i x_j = w_i w_j x'_i x'_j + u_i w_j x'_j + u_j w_i x'_i + u_i u_j$. La mise à l'échelle est donc une transforma-

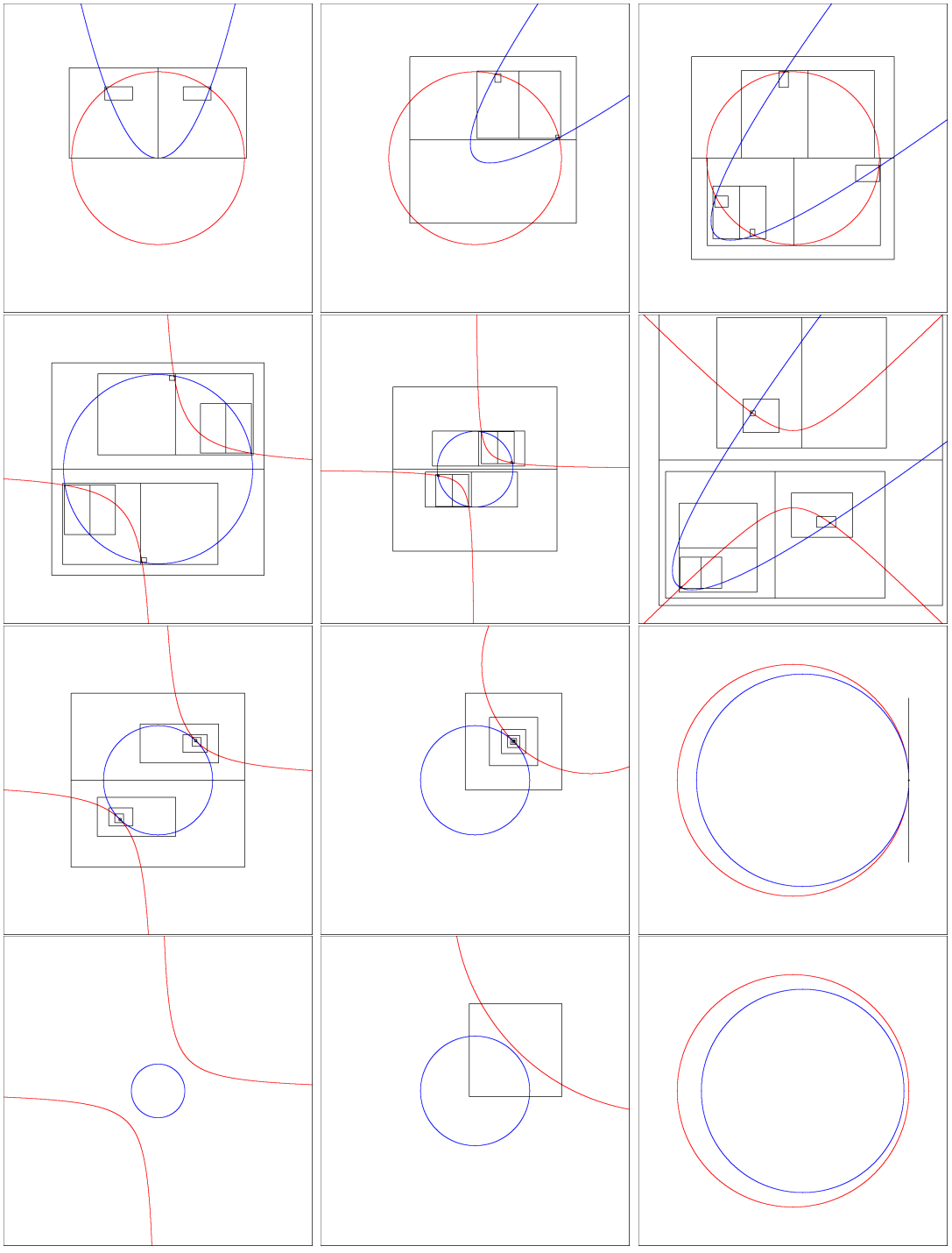

FIGURE 5.5 – Chaque figure montre tous les pavés rectangulaires, réduits ou coupés en 2. Deux premières rangées : la convergence est super quadratique autour d'un point solution non singulier. Troisième rangée : la convergence est linéaire pour les points d'intersection singuliers (aux contacts tangentiels entre deux courbes) ; cependant chaque côté du pavé est divisé par plus de 2 : autrement, le pavé serait divisé en deux. Dernière rangée : les pavés sans racine sont très vite détectés, une ou deux réductions donnant le pavé vide.

tion linéaire (affine, plus précisément) dans l'espace $\mathbb{R}^N$ des monômes, les variables de programmation linéaire.

Une autre méthode est possible ; elle adapte le polytope de Bernstein au pavé des x_i et laisse inchangées les équations et inéquations algébriques du système : $f(x) = 0, g(x) \leq 0$. Pour un pavé $x_i = [u_i, v_i]$, les inéquations des hyperplans du polytope de Bernstein sont modifiées comme suit :

$B_0^{(2)}(x_i) = (1-x_i)^2 \geq 0$ devient : $(v_i-x_i)^2 = x_i^2 - 2v_ix_i + v_i^2 = q_i - 2v_ix_i + v_i^2 \geq 0$.

$B_1^{(2)}(x_i) = 2x_i(1-x_i) \geq 0$ devient : $2(x_i-u_i)(v_i-x_i) = 2(-q_i+(u_i+v_i)x_i-u_iv_i) \geq 0$.

$B_2^{(2)}(x_i) = x_i^2 \geq 0$ devient : $(x_i - u_i)^2 = q_i - 2u_ix_i + u_i^2 \geq 0$.

$B_0^{(1)}(x_i)\ B_0^{(1)}(x_j) = (1-x_i)(1-x_j) \geq 0$ devient : $(v_i-x_i)(v_j-x_j) = x_{ij} - v_ix_j - v_jx_i + v_iv_j \geq 0$.

$B_0^{(1)}(x_i)\ B_1^{(1)}(x_j) = (1-x_i)x_j \geq 0$ devient : $(v_i - x_i)(x_j - u_j) = -x_{ij} + u_jx_i + v_ix_j - v_iu_j \geq 0$.

$B_1^{(1)}(x_i)\ B_1^{(1)}(x_j) = x_ix_j \geq 0$ devient : $(x_i-u_i)(x_j-u_j) = x_{ij}-u_jx_i-u_ix_j+u_iu_j \geq 0$. De même pour les autres inéquations.

5.4.2 Imprécision numérique

Le polytope épouse si bien l'ensemble quadratique $S \subset \mathbb{R}^N$ des points de coordonnées $(X_i = x_i, X_{ij} = x_ix_j)$ que certaines racines sont perdues, suite aux erreurs d'arrondi de l'arithmétique flottante. Par exemple, pour résoudre $x^2 - x = 0$ avec $x \in [0, 1]$, la droite $y - x = 0$ est considérée (Figure 5.1) ; si l'équation de cette droite devient : $y - x = \epsilon$ avec $\epsilon > 0$ à cause d'une erreur d'arrondi, alors aussi petit que soit ϵ les deux racines sont perdues. Par concision, on ne mentionne que le principe de trois solutions pour résoudre le problème d'imprécision numérique.

La première et la plus simple est le recours à une arithmétique rationnelle exacte (éventuellement paresseuse [86]). Cette solution a été programmée par D. Michelucci et fonctionne, mais elle est bien sûr très lente. Il est possible d'accélérer un peu cette méthode en arrondissant vers l'extérieur les bornes des intervalles des inconnues x_i : ainsi, si x_i est dans l'intervalle $[5001/10000, 10029/10031]$, cet intervalle peut être arrondi sur $[1/2, 1]$: ce dernier n'est guère plus grand et utilise des rationnels bien plus courts (avec moins de chiffres).

La seconde solution recourt à l'arithmétique d'intervalles [5, 85] ; justement la recherche sur la programmation linéaire avec intervalles ou avec des tolérances est très active actuellement.

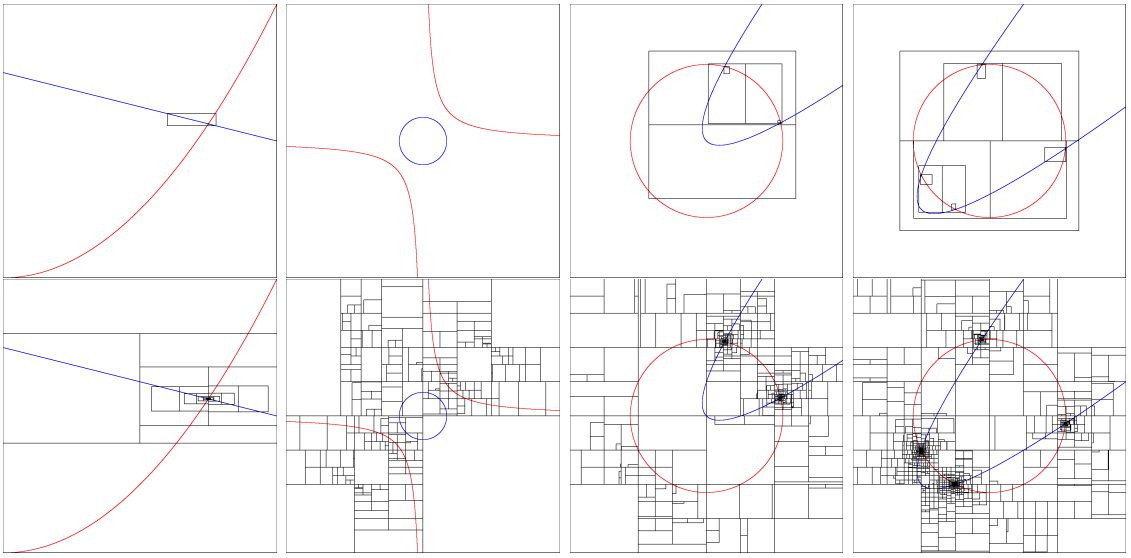

FIGURE 5.6 – Ces exemples 2D de systèmes à 2 équations et 2 inconnues sont les mêmes en haut et en bas. En haut, les pavés sont réduits avec le nouveau solveur, utilisant le polytope de Bernstein. En bas, les pavés sont réduits avec un solveur de type Newton par intervalles. Clairement, le nouveau solveur détecte bien plus vite les pavés sans racine, et converge avec moins de subdivisions. Sur ces exemples en faible dimension, le solveur classique et le nouveau solveur ont des performances très voisines.

C. Fünfzig *et al.* ont utilisé une troisième solution, détaillée dans [83]. L'analyse due à Wilkinson de la propagation d'erreur dans la résolution de systèmes linéaires permet de majorer les erreurs relatives et absolues. De plus, cette solution a l'avantage d'être compatible avec les nombreux solveurs de programmation linéaire existants, qui utilisent l'arithmétique flottante, tels SoPlex 1.4.1 (qui est utilisé dans [83]), GLPK, etc.

5.5 Conclusion

Dans cette deuxième partie, nous avons introduit une nouvelle notion mathématique appelée "Polytope de Bernstein" et présenté une méthode de réduction en temps polynomial pour surmonter la difficulté due au coût exponentiel de la représentation des polynômes multivariés dans la base tensorielle de Bernstein. Le solveur obtenu ne se réfère plus à la méthode de Newton par intervalles, ni même aux bases de Bernstein. Ce solveur a déjà été utilisé pour résoudre des contraintes géométriques en 3D : la plateforme de Gough-Stewart (ou octaèdre), le calcul des droites tangentes à quatre sphères données, le calcul de pavages de cercles, etc. Ce dernier problème génère des systèmes de taille arbitrairement grande, qui sont insolubles avec le type classique des solveurs fondés sur les bases tensorielles de Bernstein.

Des implantations GPU de ce solveur sont envisageables : en extrapolant le gain en performances que permettent les cartes GPU sur des programmes numériques voisins, elles pourraient résoudre en temps réel des systèmes de plusieurs dizaines d'inconnues.

Chapitre 6

Conclusions et perspectives

6.1 Synthèse

Cette thèse étudie la faisabilité d'un solveur résolvant des systèmes de contraintes définissant des objets géométriques et pose les problèmes et les difficultés qu'un tel solveur peut rencontrer.

Après une introduction, le chapitre 1 dresse un état de l'art dans le domaine de la modélisation géométrique de courbes et surfaces par contraintes ; nous avons introduit les différentes méthodes de la modélisation de courbes et surfaces et les différentes méthodes de résolution et de décomposition de contraintes en C.A.O.

Le chapitre 2 présente l'arithmétique par intervalles et quelques variantes limitant la surestimation, après avoir décrit le codage des nombres en virgule flottante dans un ordinateur et les problèmes inhérents à l'arithmétique des ordinateurs.

Le chapitre 3 propose un ensemble de méthodes basées sur l'arithmétique d'intervalles pour calculer et tracer des courbes algébriques. Dans un premier temps, nous avons présenté quelques méthodes parmi celles déjà existantes, à savoir l'arithmétique d'intervalles dite «naïve», la forme de Taubin et l'arithmétique affine. Nous avons démontré que la méthode de Bernstein-de Casteljau produisait des résultats visuellement bien supérieurs aux précédents, grâce à l'algorithme de Casteljau qui donne une subdivision de l'espace quasi-optimale pour une fonction exprimée dans la base de bernstein.

Le chapitre 4 présente un solveur classique fondé sur les bases de Bernstein ; ce solveur ne peut traiter que des petits systèmes, avec 6 à 7 inconnues au plus, comme il en apparaît en synthèse d'images, par exemple pour le lancer de rayons sur des surfaces paramétriques.

Le chapitre 5 présente un nouveau type de solveur : il évite cette limitation en définissant le polytope de Bernstein et en recourant à la programmation linéaire. Ce deuxième type de solveur est utilisable sur des systèmes de taille arbitraire.

6.2 Perspectives

Plusieurs pistes sont à envisager dans l'optique de travaux futurs :

- Tout d'abord la méthode de Bernstein-de Casteljau, bien que visuellement concluante dans ces résultats, est très coûteuse en temps et en espace mémoire à cause de ses appels récursifs lors de la subdivision, a fortiori en 3D.
- Une extension de la méthode aux fonctions non-algébriques afin de tracer des courbes et des surfaces.
- L'utilisation de la base de Bernstein dés le début du processus, pour exprimer les fonctions algébriques ainsi que les contraintes géométriques, pourrait être envisagée et étudiée.
- L'ajout de nouvelles contraintes, comme les contraintes de symétrie ou le passage de la surface par des lignes de caractère non echantillonnés, sont à envisager afin d'offrir un plus grand choix de contraintes à l'utilisateur qui veut créer sa forme.
- Enfin, il reste à développer une version en dimension trois de la méthode proposée surtout pour les équations non polynomiales, par exemple les fonctions transcendantes, cos, sin, ...

6.3 Conclusion

Cette étude constitue une base de travail sur la modélisation de courbes et de surfaces algébriques définies par des contraintes géométriques.
Ce document établit en premier lieu un état de l'art détaillé dans le domaine de la modélisation de courbes et de surfaces par contraintes.
Ensuite, l'apport principal de ce travail de thèse est articulé autour de deux points : d'une part le traitement de contraintes géométriques appliquées à des courbes et surfaces algébriques, et d'autre part la présentation d'une nouvelle méthode de calcul dont l'efficacité a été démontrée dans le développement du document, en dépit d'un coût élevé en temps et en espace mémoire.

Bibliographie

[1] N. Revol. Introduction à l'arithmétique par intervalles. *Ecole des jeunes chercheurs en algorithmique et calcul formel, Grenoble*, 2004.

[2] C. Jermann, D. Michelucci, and P. Schreck. Modélisation géométrique par contraintes. In Bernard Péroche and Dominique Bechmann, editors, *Informatique graphique, modélisation géométrique et animation*, pages 185–214. Hermès - Lavoisier, 2007.

[3] D. Michelucci. Solving geometric constraints by homotopy. *IEEE Trans on Visualization and Computer Graphics*, 2 :28–34, 1996.

[4] A. Sommese and C. Wampler. *Numerical solution of polynomial systems arising in engineering and science.* World Scientific Press, Singapore, 2005.

[5] R.B. Kearfott. *Rigorous Global Search : Continuous Problems.* Kluwer, Dordrecht, Netherlands, 1996.

[6] S. Ait Aoudia and S. Foufou. A 2D geometric constraint solver using a graph reduction method. *Advances in Engineering Software*, pages 1187–1194, 2010.

[7] J. Garloff and A.P. Smith. Investigation of a subdivision based algorithm for solving systems of polynomial equations. *Journal of nonlinear analysis : Series A Theory and Methods*, 47(1) :167–178, 2001.

[8] B. Mourrain and J-P. Pavone. Subdivision methods for solving polynomial equations. Technical Report RR-5658, INRIA, August 2005.

[9] G. Elber and M-S. Kim. Geometric constraint solver using multivariate rational spline functions. In *SMA'01 : Proc. of the 6th ACM Symp. on Solid Modeling and Applications*, pages 1–10, New York, NY, USA, 2001. ACM Press.

[10] M. Reuter, T.S. Mikkelsen, E.C. Sherbrooke, T. Maekawa, and N.M. Patrikalakis. Solving nonlinear polynomial systems in the barycentric bernstein basis. *Vis. Comput.*, 24(3) :187–200, 2008.

[11] D. Michelucci and S. Foufou. Bernstein basis for interval analysis : application to geometric constraints systems solving. In Bruguera and Daumas, editors, *8th Conference on Real Numbers and Computers*, pages 37–46, July 2008.

[12] R. Martin, H. Shou, I. Voiculescu, A. Bowyer, and G. Wang. Comparison of interval methods for plotting algebraic curves. *Computer Aided Geometric Design*, 19(7) :553–587, 2002.

[13] D. Michelucci, S. Foufou, L. Lamarque, and P. Schreck. Geometric constraints solving : some tracks. In *ACM Symp. on Solid and Physical Modelling*, pages 185–196, 2006.

[14] Vasek Chvatal. *Linear Programming (Series of Books in the Mathematical Sciences)*. W. H. Freeman, September 1983.

[15] A. Moussaoui. *Modélisation géométrique par contraintes*. PhD thesis, Institut National de formation en Informatique, 2000.

[16] S. Ait Aoudia. *Modélisation géométriques par contraintes : Quelques méthodes de résolution*. PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Étienne, 1994.

[17] R. Maculet and M. Daniel. Conception, modélisation géométrique et contraintes en cao : Une synthèse. *Revue d'Intelligence Artificielle*, 18(5-6) :619–645, 1982.

[18] C.M. Hoffmann and R. Joan-Arinyo. Parametric modeling. *Handbook of Computer Aided geometric Design, edition North holland*, 2001.

[19] J. Chung and M. Shussel. Comparison of varational and parametric design. *SME AutoFact'89 conference, Detroit, Michigan, USA*, 1989.

[20] D. Lesage. *Un modèle dynamique de spécification d'ingénierie basé sue une approche de géométrie varationnelle*. PhD thesis, Institut National Polytechnique de Grenoble, 2002.

[21] V.C. Lin, D.C. Gossard, and R.A. Light. Varational geometry in computer aided design. *ACMcomputer Graphig*, 15(3) :171–177, 1981.

[22] M. Fontan, F. Giannini, and F. Meirana. Free from features for aesthetic design. *International journal of Shape modelling*, 6(2) :273–302, 1982.

[23] J.P. Pernot, S. Guillet, J.C. Leon, B. Falcidieno, and F. Giannini. Multi-minimisations for shape control of fully free from deformation features. *Shape Modeling Conference*, 6(2) :273–302, 2004.

[24] J.J. Shah. A schema for cad-capp integration. *Automation Systems Laboratory, General Electric Corp. Research and Development Center, Schenectady, NY*, 1986.

[25] P. Marks. Whai do solid model need ? *Machine Design*, 12, 1987.

[26] J.J. Shah, G. Balakershnan, M.T. Rogers, and S.D. Urban. Comparative study of procedural and declarative feature based geometric modelling. *IFIP, Valenciennes*, pages 647–671, 1994.

[27] M. Lucas, D. Martin, P. Martin, and D. Plemenos. Le projet explo-formes, quelques pas vers la modélisation déclarative de formes. *GROPLAN*, 1989.

[28] P. Martin and D. Martin. An expert system for polyhedra modelling. *EUROGRAPHICS'88*, 1988.

[29] D. Chauvat. *Le projet VoluFormes : un exemple de modélisation déclarative avec contrôle spatial*. PhD thesis, Universite de Nantes, 1994.

[30] D. Plemenos. La modélisation déclaratives en synthèse d'images, tendances et perspectives. *MSI 94-03*, 1994.

[31] V. Gaildrat. *Modélisation déclarative d'environnements virtuels :Création de scènes et de formes complexes par l'énoncé de propriétes et l'emploi d'interactions gestuelles*. PhD thesis, Universite de Toulouse, 2003.

[32] M. Daniel. Declarative approach of curve modeling. *Third International Conference Curves and Surfaces*, 1996.

[33] M. Daniel and M. Lucas. Towards declarative geometric modeling in mechanics. *IDDME'96, Integrated Design and Manifacturing*, pages 427–436, 1997.

[34] R.L. Greca. *Approche déclarative de la modélisation de surfaces*. PhD thesis, Université de la Méditérranée, 2005.

[35] M. Lucas and E. Desmontils. Les modeleurs déclaratifs. *Revue internationale de CFAO et d'informatique graphique*, 10(6) :559–586, 1995.

[36] E. Desmontils and D. Pacholczyk. Vers un traitemeent linguistique des propriétés en modélisation déclarative. *Revue internationale de CFAO et d'informatique graphique*, 12(4) :351–371, 1997.

[37] M. Daniel. Conception, modélisation géometrique et contraintes en c.a.o : Une synthèse. *Rapport de recherche LSIS-2003-2005*.

[38] D. Muchelucci S. Foufou and J.P. Jurzak. Numericale decomposition of geometric constraints. *In Proceedings of the 2005 ACM symposium on solid and physical modeling*, pages 143–151, 2005.

[39] S.C. Chou, W.F. Schelter, and J.G. Yang. Characteristic sets and grôbner bases in geometry theorem proving. In *Proceedings of the workshop on computer aided geometry reasoning*, pages 29–56, INRIA, France, 1987.

[40] W. Wu. Basic principales of mechanical theorem proving in elementary geometries. *Automated Reasoning*, 2 :221–254, 1986.

[41] X-S. Gao and S-C. Chou. Solving geomtric constraint systems. ii. a symbolic approach and decision if rc-constructibility. *Computer Aided Design*, 30(2) :115–122, 1998.

[42] D. Bondyfalat. *Interaction entre symbolique et numérique : application à la vision artificielle*. PhD thesis, Université de Nice sophia antipolis, 2000.

[43] R. Latham and A. Middleditch. Connectivity analysis : a tool for processing geometric constraint. *Computer Aided Design*, 28(11) :917–928, 1996.

[44] C. Nguyen and J-C. Lafon. Constraint-based design of exact parameterised 3d solids. In *Proceedings of the CSG-96, Set theoretic solid Modeling Techniques and applications, editions Ltd,*, pages 49–63, Winchester, UK, 1996.

[45] J.F. Dufour, P. Mathis, and P. Scherck. Geometric construction by assembling solved subfigures. *Artificial Intelligence*, 99(1) :73–119, 1998.

[46] R.A. Light and D.C. Gossard. Modification of geometric models through varoational geometry. *Computer Aided Design*, 14(4) :209–214, 1982.

[47] G. Nelson and Juno. A constraint-based graphic system. *ACM Sigraph conference Proceeding Graphig*, 1985.

[48] D. Michelucci and H. lamure. Résolution de contraintes géométriques par homotopie. *Actes de AFIG*, 1994.

[49] H. Lamure and D. Michelucci. Solving geometric constraints by homotopy. In *Proceedings of the third symposium on solid modeling and applications, editions ACMPress*, pages 263–269, 1995.

[50] C.Y. Hsu and B. Brüderlin. A degree-of-freedom graph approch. *Geometric Modeling : theory and Prectice*, pages 132–155, 1997.

[51] S. Ait-Aoudia, R. Jegou, and D. Michelucci. Reduction of constraint systems. *Proceding of the Compugraphics Conference, Alvor, Portugal*, pages 83–92, 1993.

[52] N. Revol. Introduction à l'arithmétique par intervalle. 2001.

[53] L. Lamarque. *Modélisation géométrique et arithmétique par intervalles*. PhD thesis, faculté des sciences et techniques,Dijon, 2006.

[54] J.C. Burkill. Functions of intervals. *Proc. London Maths*, 22 :275–336, 1924.

[55] R. Young. The algebra of many-valued quantities. *Maths annale*, 104 :260–290, 1932.

[56] T. Sunaga. Theory of an interval algebra anf its application to numerical anakysis. *RAA Memoirs*, 2 :547–564, 1958.

[57] T. Sunaga. *Theory of an interval algebra anf its application to numerical anakysis*. PhD thesis, departement of computer science, Stanford University, 1963.

[58] R.E. Moore. Interval analysis. *Prentice-Hall, Englewood Cliffs, N.J*, 1966.

[59] D. Menégaux. *Modélisation de courbes et de surfaces algébriques par contraintes géométrique*. PhD thesis, faculté des sciences et techniques,Dijon, 2006.

[60] A. Neumaier. The wrapping effect, ellipsoid arithmetic, stability and confidence regions. *Computing Supplementum*, 9 :175–190, 1993.

[61] E. Baumann. Optimal centred forms. *BIT*, 28(1) :80–87, 1988.

[62] D. Michelucci, S. Foufou, L. Lamarque, and D. Menegaux. Bernstein based arithmetic featuring de casteljau. *CCCG*, pages 215–218, 2002.

[63] G. Taubin. An accurate algorithm for rasterizing algebraic curves. *Secon symp. on solid modeling and applications ACM/ IEEE*, pages 221–230, 1993.

[64] G. Taubin. Rasterizing algebraic curves and surfaces. *IEEE computer graphics and applications*, 14(2) :14–23, 1994.

[65] A. Andrade, D. Comba, J. Luiz, J. Stolfi, and M. Vinicius. Affine arithmetic. 1993.

[66] R.E. Moore. The automatic analysis and control of error in digital computation based on the use of interval numbers. *L.B Rall (ed), error in digital computation ? Wiley , New york*, 1 :61–130, 1965.

[67] L.H. Figueiredo and J. Stolfi. Adaptive enumeration of implicit surfaces with affine arithmetic. *Implicit Surfaces'95, Grenoble*, pages 161–170, 1995.

[68] R. Martin, H. Shou, I. Voiculescu, A. Bowyer, and G. Wang. Comparison of interval methods for plotting algebraic curves. *Computer Aided Geometric Design*, 19(7) :553–587, 2002.

[69] G. Farin. Curves and surfaces for computer aided geometric design. *Academic Press,4 edition, New york*, 1996.

[70] C. Hu, N. Patrikalakis, and X. Ye. Robust interval solid modeling. part 1 : representation. part 2 : Boundary evaluation. *Computer Aided Design*, 28(10) :807–817, 819–830, 1996.

[71] E. Sherbrooke and N. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design*, 10(5) :379–405, 1993.

[72] J. Garloff and A. Smith. Investigation of a subdivision based algorithm for solving systems of polynomial equations. *Journal nonlinear analysis : Series A theory and methods*, 47(1) :167–178, 2001.

[73] P. Nataraj and K. Kotecha. Higher order convergence for multidimensional functions with a new taylor-bernstein form as inclusion function. *Reliable Computing*, 9(3) :185–203, 2003.

[74] P. Nataraj and K. Kotecha. Global optimization with higher order inclusion function forms- part 1 : A combined taylor-bernstein form. *Reliable Computing*, 10(1) :27–44, 2004.

[75] B. Mourrain and F. Rouillier. Bernstein's basis and real root isolation. *Technical report 5149, INRIA Rocquencourt*, 2004.

[76] G. Morin and R. Goldman. Trimming analytic functions using right side poisson. *Computer Graphics*, 33(11) :813–824, 2001.

[77] COPRIN. Alias-c. Technical report.

[78] G. Farin. *Curves and Surfaces for CAGD : A Practical Guide.* Academic Press Professional, Inc., San Diego, CA, 1988.

[79] D. Michelucci and S. Foufou. Interval-based tracing of strange attractors. *Int. J. Comput. Geometry Appl.*, 16(1) :27–40, 2006.

[80] A. Paiva, L.H. de Figueiredo, and J. Stolfi. Robust visualization of strange attractors using affine arithmetic. *Computers & Graphics*, 30(6) :1020–1026, 2006.

[81] R.L. Rivest T.H. Cormen, C.E. Leiserson and C. Stein. *Introduction to Algorithms. Section 33.3, pp. 947-957 : Finding the convex hull.* MIT Press and McGraw-Hill, 2nd edition edition, 2001.

[82] F.P. Preparata and S.J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM*, 20(2) :87–93, 1977.

[83] C. Fünfzig, D. Michelucci, and S. Foufou. Nonlinear System Solver in Floating-Point Arithmetic using LP Reduction. In *Proc. of the ACM Symp. on Solid and Physical Modeling, San-Francisco, California, october 5-8*, pages 123–134, 2009.

[84] G.B. Dantzing. *A history of scientific computing*, chapter Origins of the simplex method, pages 141–151. Reading, Ma, USA, 1990.

[85] O. Beaumont. *Algorithmique pour les intervalles.* PhD thesis, IRISA, projet Aladin, 1999.

[86] D. Michelucci and J-M. Moreau. Lazy Arithmetic. *IEEE Transactions on Computers*, 46(9) :961–975, September 1997.

Résumé

Aujourd'hui plusieurs applications de l'informatique graphique ou géométrique nécessitent la résolution de systèmes non linéaires.
Le calcul des images de synthèse par lancer de rayons nécessite le calcul des points d'intersection entre des rayons (des demi-droites) et des surfaces définies par une ou plusieurs équations algébriques.

La modélisation géométrique, et la Conception et Fabrication Assistées par Ordinateur (CFAO) ont besoin de calculer les points d'intersection entre de telles surfaces. Il en résulte des systèmes d'équations algébriques de petites tailles. Enfin, tous les modeleurs géométriques utilisés en CFAO fournissent aujourd'hui la possibilité de modéliser des objets géométriques, ou de dimensionner des pièces, par un ensemble de contraintes géométriques.
La résolution de ces contraintes géométriques nécessite la résolution de systèmes d'équations algébriques non linéaires. Les sous systèmes irréductibles peuvent être de grande taille (plus d'une dizaine d'inconnues et d'équations) et sont résolus par des méthodes numériques : citons l'itération de Newton-Raphson, l'homotopie (ou continuation), les méthodes de Newton par intervalles, et les solveurs utilisant les bases tensorielles de Bernstein ou d'autres bases géométriques.

Les bases tensorielles de Bernstein permettent de calculer de bons encadrements des valeurs des polynômes sur des pavés, et de résoudre les systèmes polynomiaux rencontrés en synthèse d'images, en modélisation géométrique, et en résolution de contraintes géométriques.

Cette thèse présente deux types de solveurs. Le premier est classique fondé sur les bases de Bernstein et limité aux petits systèmes de 6 ou 7 inconnues au plus, comme il apparaît en synthèse d'images, par exemple pour le lancer de rayons sur des surfaces paramètriques.
Le second est nouveau,il évite cette limitation en définissant le polytope de Bernstein et en recourant à la programmation linéaire. Ce deuxième type de solveurs est utilisable sur des systèmes de taille arbitraire.

Mots clés
Analyse d'intevalle, Bases de Bernstein, Arithmétique d'intervalle, Solveur, Courbe et surface de Béziers, Enveloppe convrxe, Bissection, Polynôme uni-varié, Polynôme multi-varié